Fachhochschule Köln, Abteilung Gummersbach

Diplomarbeit

zur Erlangung des akademischen Grades eines Diplom-Informatikers

Fachbereich Informatik, Studiengang Wirtschaftsinformatik

Objektorientierte Ansätze in einer relationalen Datenbankumgebung

Fallbeispiel "Vertriebsinformationssystem"

Thomas Barthels

Betreuer:

Prof. Dr. Heide Faeskorn-Woyke

Dirk Kockmann

Januar 1999

Inhalt:

An dieser Stelle werden wesentliche Begriffe im Zusammenhang mit der Diplomarbeit erläutert. Es werden die Vor- und Nachteile der relationalen und objektorientierten Sicht der Datenhaltung aufgeführt.

1. Einführung in die Thematik

1.1. Von der Datei zur Datenbank

Die von einem Programm zur Laufzeit verwendeten und erzeugten Daten sind ohne spezielle Vorkehrungen beim Beenden des Programms verloren. Für die meisten Anwendungen ist diese Tatsache aber sicherlich unerwünscht. Daten, die man also unabhängig vom aktuellen Programmablauf in einer Anwendung zur Verfügung haben muß, müssen entsprechend auf Speichermedien ausgelagert werden, um dem Programm sekundäre oder historische Daten zur Verfügung stellen zu können.

Als ein in diesem Zusammenhang verbreitetes Vorgehen findet man oft die Möglichkeit erzeugte Daten in einer sogenannten "Datei" abspeichern zu können. Der Aufbau der Daten in einer Datei hängt meistens von dem jeweiligen Programm ab, wenn es sich nicht um einen Standarddateityp handelt. Das hat zur Folge, daß die Daten dann auch nur von dem ursprünglichen Programm genutzt werden können.

Besonders in der unternehmerischen Praxis ergibt sich aber oft ein sehr viel komplexerer Anspruch an Daten. Sie sollen zum Beispiel nicht nur einem Programm zur Verfügung stehen, da es sich vielleicht um zentrale Daten handelt, die an verschiedenen Stellen zur Verfügung stehen müssen, aber im Gegensatz dazu nur an einer Stelle gepflegt werden sollen. Wesentliche Kriterien in diesem Zusammenhang sind die Ansprüche auf Vermeidung von Redundanzen und Einhaltung der Konsistenz eines Datenbestandes. Werden an unterschiedlichen Stellen in einem Unternehmen entsprechende Daten dezentral erfaßt und verarbeitet, lassen sich diese Ansprüche ohne zusätzliche Logistik kaum realisieren.

Es ergibt sich die Notwendigkeit für eine integrierte Verwaltung der zentralen Daten eines Unternehmens. Ein wesentliches Merkmal dieser integrierten Verwaltung ist es, daß nicht jedes Programm autonom seine Datenelemente und –strukturen festlegt, sondern das diese unabhängig vorhanden sind.

Dies nun kann ein modernes Datenbanksystem leisten. Es bietet Raum für persistente Daten, die unabhängig, flexibel und geschützt abgelegt werden können. Die ebenfalls enthaltenen "Regeln" gewähren mehreren Benutzern gezielten Zugriff.

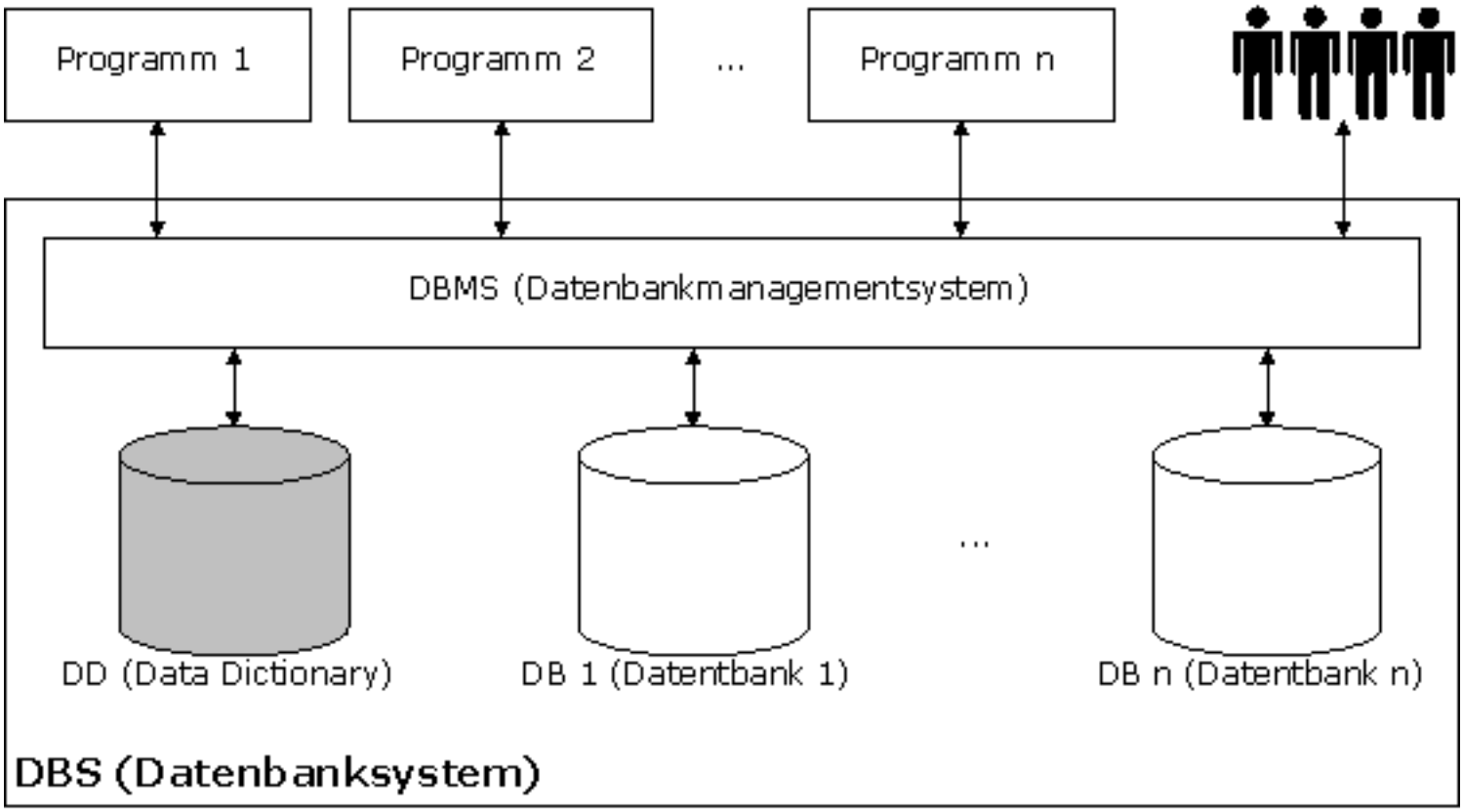


Abbildung 1-1

Die Datenbanken enthalten die Daten, das Data Dictionary enthält das Datenbankschema, und das Datenbankmanagementsystem verwaltet die Zugriffe auf den Datenbestand.

Die Daten selber werden hinsichtlich ihrer Bedeutung im Datenmodell beschrieben. Es legt die Eigenschaften, Struktur, Konsistenzbedingungen und Operationen fest. Syntax und Semantik eines Datenmodells werden durch eine Definitionssprache (DL) und eine Manipulationssprache (ML) bestimmt. Das Datenbankschema beschreibt einen konkreten Einsatzfall.

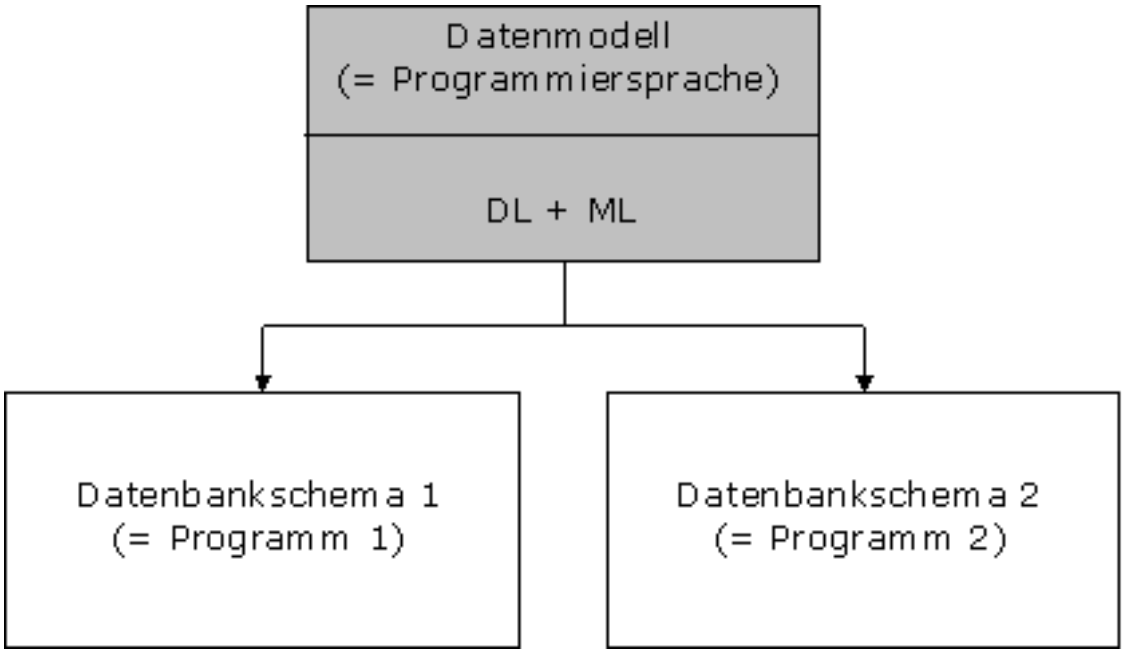


Abbildung 1-2

1.2. Relationales und objektorientiertes Datenmodell

Heute werden in der Praxis überwiegend relationale und objektorientierte Datenmodelle verwendet:

Relationale Datenbanksysteme speichern Daten in Tabellenform. Die bei der Normalisierung gefundenen Entitätsmengen werden auf entsprechende Tabellen abgebildet. Jede Spalte einer Tabelle repräsentiert ein Attribut der Entitätsmenge. In jeder Zeile kann eine Entität der Entitätsmenge gespeichert werden (Tupel). Die Definition des logischen Schemas geschieht mit Hilfe der Datendefinitionssprache (DDL), die das DBMS zur Verfügung stellt. Als Standard hat sich die Sprache SQL (Structured Query Language) etabliert. Um in der erzeugten Struktur Daten zu manipulieren existiert zusätzlich eine Datenmanipulationssprache (DML). Da diese normalerweise über keine eigenen Kontrollstrukturen verfügt, ist sie oft in eine höhere Programmiersprache eingebettet. Als DML wird in der Regel ebenfalls SQL verwendet.

Objektorientierte Datenbanksysteme speichern Objekte in unveränderter Form in der Datenbank. Als Ausgangspunkt wird ein OOA-Modell (Object Oriented Analysis-Model) verwendet. Die ODMG (Object Database Management Group) ist bemüht einen Standard für objektorientierte Datenbanksysteme festzulegen. Sie verwendet als Definitionssprache ODL (Object Definition Language). Jede Klasse des OOA-Modells wird in ODL durch eine Schnittstellendeklaration beschrieben. Die Menge aller Objekte einer Klasse bezeichnet man als Ausdehnung (Extent). Über die Extent-Bezeichnung kann man auf alle Objekte zugreifen, die von einer Klasse erzeugt oder gelöscht worden sind. Beziehungen zwischen Klassen (Assoziationen und Aggregationen) werden durch "Relationships" beschrieben. Die Angabe "Inverse" sorgt dafür, daß Beziehungen in beiden Richtungen durchlaufen werden können. Die möglichen Operationen werden als Signatur angegeben. Sie enthalten Ein- und Ausgabeparameter einschließlich ihrer Typen und gegebenenfalls Ausnahmebehandlungen (Exceptions). Anfragen an eine bereits gefüllte Datenbank können mit der Abfragesprache OQL (Object Query Language) durchgeführt werden. OQL besitzt im Gegensatz zu SQL keine Einfüge- oder Änderungsoperationen. Diese müssen innerhalb von Anwendungsprogrammen implementiert werden. Die Definition von "Sichten" auf die Daten kann durch vordefinierte Anfragen erreicht werden. Sie sind aber eine reine Erweiterung der Anfragesprache und können somit nicht im Datenbankschema definiert werden.

1.3. Wesentliche Unterschiede zwischen relationalen und objektorientierten Datenbanken

1.3.1. Wertidentität und Objektidentität

Im relationalen Datenmodell werden Datensätze durch Schlüssel identifiziert. Über diesen Schlüssel kann man auf die jeweiligen Datensätze verweisen. Nimmt man ihn zum Beispiel in anderen Tabellen

als Fremdschlüssel auf, kann man mit Hilfe dieser Verweise komplexe Relationen verwirklichen. Problematisch ist es, daß die Schlüsselattribute von Aktualisierungs-Operationen grundsätzlich wie jedes andere Attribut behandelt werden. Ohne Vorkehrungen in der Anwendungsumgebung können Änderungen bei Schlüsselattributen leicht zu Inkonsistenzen führen.

In objektorientierten Datenmodellen herrscht eine strikte Trennung des in der Datenbank dargestellten Objekts von seinen Werten. Jedes Objekt hat eine unveränderliche Identität (OID) und unabhängige Attributwerte, die es beschreiben. Durch Objekt-Identitäten kann man dynamisch willkürlich zusammengesetzte oder komplexe Objekte konstruieren. Für Objekte gibt es sowohl die Möglichkeit der Identität als auch die Möglichkeit der Gleichheit. Zwei Objekte sind gleich, wenn alle relevanten Attribute gleich sind und alle Attribute, die Objekte sind, rekursiv gleich sind. Sie unterscheiden sich also nur in ihrer OID. Alle Objekte können von anderen Objekten referenziert werden. Änderungen an Objekten ergeben wieder "dasselbe" Objekt.

Man kann ein wertbasiertes System, wie es eine relationale Datenbank darstellt, um Objekt-Identitäten erweitern:

- Die Tupel müssen um Tupel-Bezeichner ergänzt werden.
- Es dürfen keine wertbasierten Referenzen zwischen Tupeln bestehen.
- Die Überprüfung der referentiellen Integrität muß implementiert sein.
- Aktualisierungs-Operationen müssen implementiert sein.
- Identitäts- und Gleichheits-Operatoren müssen implementiert sein.

1.3.2. Einfache und komplexe Objekte

In relationalen Datenbanksystemen können nur Tabellen mit Feldern fester Länge bestimmt werden. Die Datensätze belegen somit immer den gleichen Speicherplatz. Für die Attribute bedeutet dies eine Beschränkung auf die einfachen Typen (CHARACTER, INTEGER, FLOAT, DATE etc.). Das führt bei der Normalisierung der Daten beziehungsweise bei deren Handhabung zu Problemen. Läßt man die Daten in einem groben Zusammenhang bestehen (meistens Zeichenketten) ist der Zugriff auf einzelne Teile eines Attributes schwierig bis unmöglich (man handelt in diesem Fall sowieso gegen die Konzepte der Normalisierung). Trennt man Daten blind bis in ihre kleinsten Bestandteile, geht unter Umständen der logische Zusammenhang verloren. Bildet man schließlich neue Tabellen, entsteht oft erheblicher Programmieraufwand, um für den Benutzer trotzdem eine natürliche Sicht und Handhabung der Daten zu gewährleisten.

In objektorientierten Datenbanksystemen findet man hingegen Möglichkeiten der Strukturierung, wie man Sie von objektorientierten Programmiersprachen her kennt (STRUCT, ARRAY, etc.). Eine Struktur zum Beispiel besteht aus einer festen Anzahl von benannten Komponenten. Die

Komponenten enthalten ein Objekt oder ein Literal. Die Typen der Komponenten können unterschiedlich sein. Es handelt sich also dann um Typen, die aus anderen Typen konstruiert werden. Durch die Verschachtelung können komplexe Objekte modelliert werden.

1.3.3. Sichtbare und gekapselte Attribute

In einer relationalen Datenbank sind grundsätzlich alle Attribute für den Benutzer beziehungsweise für das jeweilige Anwendungsprogramm sichtbar. Die Operationen beziehen sich unmittelbar auf die Attributnamen. Da die Attributtypen elementare Typen sind, kann man sie direkt mit den entsprechenden Operationen auswerten und bearbeiten.

Die objektorientierte Logik schreibt das genaue Gegenteil vor. Die Attribute eines Objektes sind nach Außen verborgen, und man kann nur mittels bereitgestellter Operationen auf sie zugreifen. In Datenbanksystemen muß man aber oft über bestimmte Attribute auf Daten zugreifen. Deswegen ist hier die Einkapselung der Attribute auch nicht streng nach dem objektorientierten Prinzip durchgesetzt. Teile der Objektwerte können streng gekapselt werden, andere können für frei zugängliche Eigenschaften verwendet werden. Diese Überlegungen müssen beim Datenbankentwurf in einem entsprechenden Schema berücksichtigt werden.

1.3.4. Anbindung an eine Programmiersprache

Relationale Datenbanksysteme waren zuerst als alleinstehende Informationssysteme konzipiert. Alleine mit Hilfe von deklarativen Programmiersprachen (zum Beispiel SQL) wollte man die Datenbanken programmieren. Komplexe Anwendungsprogramme ließen aber bald das Fehlen von Schleifen, Rekursion, Prozeduren und ausreichenden mathematischen Operationen vermissen. Alternativ gibt es daraus folgend die Möglichkeiten, die deklarativen Programmiersprachen um die fehlenden Strukturen zu erweitern oder klassische Programmiersprachen um deklarative Komponenten zu ergänzen. In beiden Fällen ergeben sich Probleme hinsichtlich der unterschiedlichen Typmodelle (Datenkonversion), der Komplexität der Programme, der Wartbarkeit der Programme, der Verteilung der Laufzeitsysteme und des Laufzeitverhaltens.

Bei objektorientierten Datenbanksystemen liegt eine enge Anbindung an die heute meistens ebenfalls objektorientierten Programmiersprachen nahe. Man ist zur Beschreibung des Datenbankschemas nicht unbedingt auf die Schemadefinitionssprache ODL angewiesen, sondern kann zum Beispiel auf eine externe Klassendeklaration zurückgreifen, die ein erweitertes Typsystem enthält. Zur Übersetzung in ODL kann man dann einen Deklarationspräprozessor verwenden. Im Implementationsteil der Programmiersprache legt man die Aufrufe für Zugriffsfunktionen fest. In der ODMG-Spezifikation sind für die Manipulation von Datenbanken Erweiterungen für verschiedene Programmiersprachen definiert.

1.3.5. Server- und Clientorientierung

Ist eine Arbeitsumgebung in Clients und Server aufgeteilt, so befindet sich der größte Teil eines relationalen Datenbanksystems auf dem Server. SQL dient hier zur Kommunikation zwischen Client und Server. Der hauptsächliche Teil der Datenverarbeitung findet auf dem Server statt.

Im Gegensatz dazu findet bei objektorientierten Datenbanksystemen der größte Teil der Datenverarbeitung auf den Clients statt. Der Server übernimmt oft nur die Aufgaben der physischen Protokollierung, der Verwaltung der Commit-Vorgänge, der Sicherheitsprüfungen, der Sperrungen, der physischen Speicherverwaltung und teilweise der Anfrageverwaltung. Es gibt sogar Ansätze bei denen der Server nur noch die angeforderten Speicherseiten zur Verfügung stellt.

1.3.6. Daten- und Objektspeicherung

In relationalen Datenbanksystemen werden die Werte von Attributen in den jeweiligen Tabellen abgespeichert. Die Operationen zur Manipulation sind im entsprechenden DBMS enthalten und beziehen sich nicht speziell auf einzelne Tabellen.

In der objektorientierten Logik gehören zu einem Objekt sowohl dessen Attributwerte als auch die definierten Operationen. In objektorientierten Datenbanksystemen werden gleichwohl meistens nur die Attributwerte gespeichert. Die Operationen werden entsprechend in einer objektorientierten Programmiersprache implementiert. In Zukunft wird man aber wohl vermehrt dazu übergehen, auch die Operationen im objektorientierten Datenbanksystem zu integrieren.

1.3.7. Persistente und transiente Daten

Transiente Daten beenden ihre Lebensdauer zusammen mit dem Programm, von dem sie erzeugt wurden. Wenn zum Beispiel solche Daten dauerhaft in einer Datenbank gespeichert werden, werden sie zu persistenten Daten.

In relationalen Datenbanksystemen sind grundsätzlich alle in der Programmiersprache deklarierten Daten transient. Sie können nur innerhalb entsprechender Programmteile explizit in persistente Daten umgewandelt werden (Tabellen, Dateien). Der Zugriff auf temporäre Sichten mittels SQL wird nur von einigen Datenbanksystemen unterstützt.

Die objektorientierte Welt betrachtet transiente und persistente Daten als völlig gleichwertig. Eine Grenze zwischen der Programmiersprache und der Datenbank soll für den Programmierer nicht erkennbar sein. Er soll sich ausschließlich mit der Klassenstruktur beschäftigen können, ohne auf zum Beispiel Umformatierungen achten zu müssen.

1.4. Ziel der Diplomarbeit

In der Praxis trifft man heute überwiegend relationale Datenbanken an. Auf der anderen Seite hat sich bei der Softwareentwicklung die Objektorientiertheit immer mehr durchgesetzt. Dies gilt gleichermaßen für Analyse, Entwurf und Entwicklung. Unter diesen Voraussetzungen wird beschrieben, wie ein Softwareprojekt nach objektorientierten Grundsätzen durchgeführt werden kann und wie daraufhin die Schritte zur Anbindung an eine relationale Datenbank aussehen. Weiterhin werden Möglichkeiten untersucht relationale Datenbanken selber um objektorientierte Aspekte zu ergänzen.

I

In diesem Kapitel wird die Firma "Ihr Partner GmbH" vorgestellt, in welcher ich während der Diplomarbeit viele Erfahrungen gesammelt habe und von deren Mitarbeitern ich bei der Durchführung der Diplomarbeit unterstützt worden bin.

IHR PARTNER
SOFTWARE + CONSULTING

2. Vorstellung der Firma "Ihr Partner GmbH" *

2. Vorstellung der Firma "Ihr Partner GmbH"

2.1. Tätigkeitsfeld und Markt

Ihr Partner bietet mittelständischen Fertigungsunternehmen eine komplette kaufmännische Softwarelösung auf Basis von "Windows 95/98/NT" basierend auf "Microsoft SQL Server" oder "Sybase SQL Anywhere" an. Mit verschiedenen aufeinander abgestimmten Softwarebausteinen, richtet sich Ihr Partner an Kunden aus verschiedenen produzierenden Branchen (Maschinenbau, Elektronikfertigung etc.). Das Kernprodukt ist die eigene Entwicklung "Ihr Partner PPS Line SQL".

I

Die "PPS Line SQL" ist ein kostengünstiges PPS System, das auf Warenwirtschaftssysteme verschiedener Hersteller aufgesetzt werden kann. In erster Linie wird die "PPS Line SQL" heute mit der "Office Line SQL", Software des weltweiten Marktführers für kaufmännische PC Systeme "SAGE KHK", eingesetzt. Die "Office Line SQL" besteht aus den Modulen Warenwirtschaft, Rechnungswesen und Personalwesen.

Es gibt zwei Vertriebskanäle für die "PPS Line SQL":

- Projektgeschäft:

Als erfahrenes Systemhaus realisiert Ihr Partner die komplette Einführung und Koordination der neuen PPS Softwarelösung. Zu den Aufgaben gehören die Unternehmensberatung und das Consulting, die Mitarbeiterschulungen, die Installation und die Inbetriebnahme der Software, Individualprogrammierung etc.. Ihr Partner ist mit ca. 350 Installationen bundesweit der größte Anbieter für "SAGE KHK Office Line Software" und wurde als erstes Unternehmen zum "SAGE Solution Center" qualifiziert.

- Seriengeschäft:

Die "PPS Line SQL" wird über ca. 450 "SAGE KHK" Händler als Standardsoftwarelösung im deutschsprachigen Raum (Deutschland, Österreich und Schweiz) aktiv vertrieben.

2.2. Firmenentwicklung

Ihr Partner wurde von Frank Türling am 01.04.1995 als Einzelfirma gegründet. Am 15.10.95 eröffnete Frank Türling ein Büro im Technologiezentrum Jülich. Vor dem Einzug ins TZJ wurde ohne Mitarbeiter von zu Hause gearbeitet. Zum 01.02.1996 wurde der erste Mitarbeiter eingestellt.

Der erste Schritt war der Vertrieb der "SAGE KHK" Software. Mitte 1996 wurde mit der Entwicklung der eigenen Software "PPS Line SQL" begonnen. Die Software liegt heute in der Version 2.02 vor und ist ca. 75 mal in mittelständischen Unternehmen installiert worden. (Im PPS Markt gilt der Anbieter als etabliert, der 10 Systeme vermarktet hat.)

Zum 01.01.1997 wurde die Ihr Partner GmbH gegründet. Gesellschafter wurden:

- Frank Türling (75%)

- S-UBG AG (25%)

(Beteiligungsgesellschaft der Sparkassen der Wirtschaftsregion Aachen)

Stammkapital: 200 TDM

I

Geschäftsführer: Frank Türling

Heute hat Ihr Partner 22 Mitarbeiter, die Bürofläche im Technologiezentrum ist auf ca. 300qm erweitert worden.

2.3. Zukunftsaussicht

Die "PPS Line SQL" in Verbindung mit "SAGE KHK Office Line" kostet inklusive der Dienstleistungen zur Implementation ca. 50 TDM (abhängig von der Unternehmensgröße). In diesem Marktsegment ist die "PPS Line" so gut wie konkurrenzlos. Andere PPS Systeme kosten nicht selten um den Faktor 10 mehr. Aufgrund dieses attraktiven Preises und der Kooperation mit "SAGE KHK" ("Sage KHK" hat in Deutschland ca. 250.000 Installationen, weltweit ca. 1,2 Mio. Installationen), gibt es hervorragende Perspektiven und Entwicklungsmöglichkeiten.

Ca. 50.000 Unternehmen in Deutschland sind potentielle Kunden, die heute überwiegend noch keine integrierte Software für Ihre Fertigung einsetzen.

Darüber hinaus stimulieren die Einführung des EURO und die Umstellung auf das Jahr 2000 den gesamten EDV-Markt und versprechen eine positive Geschäfsentwicklung für die nächsten Jahre.

2.4. Referenzkunden

- afri-cola (Köln)
- Alcatel Kabel AG & Co. (Mönchengladbach)
- allkauf Haus GmbH (Mönchengladbach)
- CWS Lackfabrik (Düren)
- Kaufring AG (Düsseldorf)
- Lufthansa Miles & More (Frankfurt / Salzburg)
- Pilkington Solar GmbH (Köln und Gelsenkirchen)
- Post Direkt GmbH (Bonn)
- S-UBG AG (Aachen)

I

In diesem Kapitel werden die einzelnen Komponenten der "Sage KHK Office Line" vorgestellt. Anschließend wird erläutert welche Aufgaben das neue Modul "VIS SQL 1.01" erfüllen soll, und wo es integriert werden soll.

3. Das Projekt: Integration des Moduls "VIS SQL 1.01" in die "Sage KHK Office Line"

3.1. Die "Sage KHK Office Line"

Die " Sage KHK Office Line" ist modular aufgebaut und in "Microsoft Office 97" integriert. Die einzelnen Komponenten können unabhängig voneinander oder im Verbund genutzt werden. Sie bestehen aus der Finanzbuchhaltung, der Auftragsbearbeitung und der Lohn- & Gehaltsabrechnung.

- Finanzbuchhaltung:

Das Finanzbuchhaltungsmodul beinhaltet eine eigenständige Lösung für die Buchhaltung von kleinen und mittelständischen Unternehmen. Zu den wesentlichen Merkmalen gehören die Euro-Fähigkeit, die Mehr-Mandantenfähigkeit, die Fremdwährungsfähigkeit, eine interne Kostenrechnung, Automatikbuchungsfunktionen, eine integrierte Anlagenbuchhaltung und umfangreiche Möglichkeiten zur Auswertung und Bearbeitung der Daten. Es existieren Schnittstellen zu den übrigen "Sage KHK Office Line" Modulen sowie zu verschiedenen Fremdsystemen.

- Auftragsbearbeitung:

I

Die Auftragsbearbeitung ist für die Erfassung und Erledigung der standardmäßigen Vorfälle im Geschäftsleben konzipiert. Neben der Mehr-Mandantenfähigkeit bietet sie einen einheitlichen Adreßstamm, die Möglichkeit zur Verwaltung beliebig vieler Ansprechpartner, mehrwährungsfähige Kontokorrente, mehrsprachige Artikelverwaltung, mehrstufige Artikelstücklisten, hierarchische Preis- und Rabattlisten, differenzierte Preiskalkulationen, mehrsprachige Textbausteine für alle Belege, konfigurierbare Belegarten, definierbare Druckprozesse sowie vielzählige Möglichkeiten Daten auszuwerten und zu bearbeiten. Auch hier gibt es Schnittstellen zur "Sage KHK Office Line" und zu Fremdsystemen.

- Lohn- & Gehaltsabrechnung:

Das Modul Lohn & Gehaltsabrechnung ist ein Personal-verwaltungssystem zur Lohnbuchhaltung in einem Unternehmen. Auch dieses Modul ist euro- und mehr-mandantenfähig. Kernfunktionen sind hier die Möglichkeit der Zuordnung der Mitarbeiter zu verschiedenen Abrechnungskreisen, umfangreiche Möglichkeiten zur Lohnbestimmung und Berücksichtigung der gesetzlichen Abgaben. Es existieren Schnittstellen zu den "Sage KHK Office Line" Produkten sowie zu Fremdsystemen.

3.2. Die Integration des Moduls "VIS SQL 1.01"

"VIS SQL 1.01" ist in erster Linie als Ergänzung zur Auftragsbearbeitung der "Sage KHK Office Line" gedacht. Das Programm soll sowohl in die Oberfläche als auch in die Daten der Auftragsbearbeitung integriert werden.

Es handelt sich dabei um eine Erweiterung der Funktionalität der Auftragsbearbeitung, die den Vertriebsinnen- und Außendienst unterstützen soll.
In erster Linie geht es darum, daß Aktivitäten in Bezug auf Kunden erfaßt und nachgehalten werden können. Außerdem soll die Anbindung an "Microsoft Office 97" verstärkt werden, in der Weise, daß innerhalb der Auftragsbearbeitung die Zeitmanagement- und Email-Funktionen von "Microsoft Outlook", sowie Funktionen von "Microsoft Word" und "Microsoft Excel" genutzt werden können.

Nachfolgend werden wichtige Begriffe und Grundlagen für eine vertriebsunterstützende Software erläutert. Abschließend werden wichtige Leistungsmerkmale erarbeitet, die aufgrund von Gesprächen und unter Zuhilfenahme entsprechender Fachliteratur gesammelt wurden.

4. Anforderungen an CAS-Standardsoftware

4.1. Einordnung

In der elektronischen Datenverarbeitung haben sich in den letzten Jahren und Jahrzehnten Begriffe wie CAD (Computer Aided Design), CAE (Computer Aided Engineering), CAP (Computer Aided Planning), CAT (Computer Aided Testing) und CAQ (Computer Aided Quality-Assurance) immer mehr durchgesetzt. Dahinter stehen erfolgreiche Konzepte für Konstruktion, Projektierung, Arbeitsplanung, Prüfung und Qualitätssicherung. Teilweise sind diese Ansätze zu umfassenden PPS-Systemen (Produktions-Planungs- und Steuerungs-Systemen) zusammengewachsen, die ein großes Kostensenkungs- und Effizienzsteigerungs-Potential beinhalten.

Nachdem innerbetriebliche Abläufe immer stärker durch die elektronische Datenverarbeitung unterstützt werden, bemüht man sich nun auch externe Abläufe mit einzubeziehen. In Vertrieb, Verkauf und Kundenbetreuung spielen CAS-Systeme eine immer größere Rolle.

Die wichtigsten Anforderungen an ein Hilfsmittel zur Vertriebsplanung, Akquisitionsunterstützung und Erfolgskontrolle sind:

- Dezentrale und zentrale Speicherung von Kunden- und Besuchsdaten
- Terminplanung und –übersicht für Kundenkontakte, Überwachung aller Kunden/Interessenten auf geplante und nicht-geplante Termine
- Übernahme von Daten aus dem zentralen Rechner auf Vertriebs- und Außendienst-PCs
- Erfassen von Besuchsberichten beim Außen- und/oder Innendienst auf elektronischem Weg, mit der Möglichkeit von Auswertungen und Statistiken

4.2. Chancen für ein CAS-System durch aktuelle EDV-Tendenzen

Die elektronische Datenverarbeitung hat sich bis heute soweit entwickelt, daß sie nicht mehr nur der Kostensenkung und der Gestaltung von Arbeitsabläufen dient. Man kann durch einen entsprechenden Einsatz sogar strategische Vorteile am Markt erzielen.

Drei Entwicklungstendenzen hatten darauf einen wesentlichen Einfluß:

- Standardisierung:

  Die Durchsetzung von Standards in der Anwendungssoftware, der Kommunikation und den Datenbanken gewährleistet definierte Schnittstellen.

- Dezentralisierung:

  Großrechner verlieren mehr und mehr an Bedeutung. Netzwerke, Workstations und Low-Cost-Systeme bestimmen zunehmend die Computer-Landschaft.

- Vernetzung und Integration:

  Neue Wege der System-Kommunikation (Internet, BTX, ISDN etc.) ermöglichen die Datenerfassung ortsunabhängig und ohne Zeitverlust.

Eine Folge ist, daß der Computer nicht nur stationär eingesetzt wird, sondern daß vielmehr vielfältige Situationen für die Datenerfassung denkbar sind. Innen- und Außendienst können so zum Beispiel mit den gleichen Daten an unterschiedlichen Orten arbeiten. Für die Konsistenz der Daten und den notwendigen Informationsfluß ergeben sich hierdurch natürlich besondere Anforderungen.

Die Aufgaben der Verkaufsabrechnung wie Auftragsbearbeitung und Fakturierung spielen sich normalerweise weiterhin zentral ab. Die oft noch manuell durchgeführte Verkaufsdisposition mit Vertriebsplanung, Akquisitions-unterstützung und Vertriebsberichtssystem kann jetzt aber dezentral verteilt werden. Beide Bereiche sollen zu einem geschlossenen Vertriebsinformationssystem verschmelzen.

Angestrebt sind:

- Eine transparente Kunden- und Produktdarstellung mit der Möglichkeit zu Produktanalysen und Wettbewerbsbeobachtungen
- Eine schnelle und funktionierende Kooperation und Kommunikation innerhalb des gesamten Vertriebs
- Eine schnelle, flexible und kostenoptimale Abwicklung aller vertriebsspezifischen Aufgaben

Es sollen sich folgende Wettbewerbsvorteile ergeben:

- Außendienst- und Kundenkontakte erfolgen zunehmend "online" (mit möglichst geringen Vezögerungen).
- Die Abstimmung zwischen Innen- und Außendienstmitarbeitern in Hinblick auf die Kunden wird optimiert. Sie werden zu Koordinatoren zwischen den Kunden und ihrem Unternehmen.
- Der Informationsfluß im gesamten Unternehmen soll sich verbessern, so daß der Verkäufer auf die Kompetenz des Gesamtunternehmens zurückgreifen kann.
- Die Durchlaufzeiten bei der Auftragsannahme und der Auftragsweiterleitung sollen sich signifikant verkürzen.

Eine CAS-Lösung soll schließlich vollständig an ein PPS-System angebunden werden und somit Bestandteil des Konzeptes der CIM (Computer Integrated Manufacturing) werden.

4.3. Generelle Auswahlkriterien

- Verbesserung des Kundenmanagements: Besuchsvorbereitung, Kundenbearbeitung, Besuchsfrequenzen, Telefon-Aktionen, Angebotserstellung, Angebotsverfolgung, Angebotsanalysen, Umsatzkontrolle etc.
- Größere Kundennähe: Kerndaten aus den Besuchsberichten können gezielt, systematisch, vollständig und schnell erfaßt und ausgewertet werden.

- Bessere Wettbewerbsbeobachtung: Komplexität und Veränderungen haben in fast allen Märkten zugenommen. Verhalten und Aktivitäten der Wettbewerber müssen genauer, systematischer und kurzfristiger beobachtet werden.

- Gezieltere Außendienststeuerung: Verbesserung der Führung und Steuerung des Außendienstes. Objektivierung der Potentialberechnung und Zielvorgabe in den Verkaufsbezirken.

- Präzisierung der strategischen Positionierung: Vergleiche der eigenen Produkte und Strategien mit denen der Wettbewerber sollen ständig die eigene Position verbessern und zu Innovationen beitragen.

4.4. Wirtschaftlichkeit

Für den Vertrieb sind die Marktpräsenz und die Erfolgsrate Schlüsselbereiche. Der Außendienstmitarbeiter soll so lange und so oft wie möglich bei den Kunden vor Ort sein. Dabei ist es wichtig, daß er sich an die "richtigen" Kunden wendet, an die also, bei denen die Abschlußwahrscheinlichkeit sehr hoch ist.

Die dezentrale Datenverarbeitung soll den Aufwand für Verwaltungstätigkeiten (Besuchsvorbereitung, Besuchsnachbearbeitung und interne Tätigkeiten) reduzieren. Die notwendigen Daten für eine optimale, zeit- und sachgerechte Kundenbetreuung müssen also zur Verfügung gestellt werden. Der erwirtschaftete Zeitvorteil kann für die eigentlichen Verkaufstätigkeiten genutzt werden.

Arbeitszeitanalysen eines typischen Außendienstmitarbeiters ergeben, daß 15% der Arbeitszeit eingespart werden können. Davon können

- 50% in der Besuchsvorbereitung eingespart werden. Datensammlung, Such- und Auswertungsvorgänge sollen automatisiert werden.

- 20% durch eine automatisierte Erstellung von Besuchsberichten und Statistiken eingespart werden.

- 30 % durch ein asynchrones Kommunikationssystem (Email, Mailbox-Systeme etc.) eingespart werden. Die direkte Kommunikation zwischen Zentrale und Außendienstmitarbeiter (meistens Telefon) macht oft vergebliche Kommunikationsversuche nötig ("besetzt", "nicht am Platz" etc.).

Das Kriterium "Höhere Erfolgsrate pro Besuch" wird sicherlich durch eine gute Besuchsvorbereitung, aktuelle Beobachtung der Kundenentwicklung, laufende und gezielte Auswertung aller Kundenreaktionen sowie frühzeitiger Abweichungsanalysen positiv beeinflußt. Aktuelle Daten, die übersichtlich und schnell zur Verfügung stehen, spielen auch hier eine entscheidende Rolle.

4.5. Einführung der Software

Die Vorteile einer noch so guten Software können erst dann genutzt werden, wenn sie überhaupt in einer vorhandenen Umgebung einsetzbar ist und von den Benutzern akzeptiert wird.

Technische Aspekte:

- Softwareergonomie: Bei einer grafischen Benutzeroberfläche ist darauf zu achten, daß die dort verwendeten Konzepte (Buttons, Scrollbars, Menüs etc.) in der Software gleichbedeutend verwendet werden. Bei der Integration in ein anderes Anwendungssystem sind zusätzlich die dort verwendeten Ansätze (zum Beispiel Dialoggestaltung) zu berücksichtigen.

- Anforderungen an die Hardware: Das Laufzeitverhalten des Programms sollte auf keinen Fall den normalen Arbeitsablauf beeinträchtigen oder unterbrechen. Die Reaktionszeiten bei einer standardmäßigen Dialogführung sollten im zu erwartetenden Rahmen bleiben (bezogen auf eine zeitgemäße Hardwareumgebung).

Menschliche Aspekte:

- Akzeptanzprobleme: Meistens sind Menschen eher mißtrauisch gegenüber Neuerungen in ihrer gewohnten Umgebung eingestellt. Programme, die sichtbar mit Funktionen überladen sind (Menüs, Buttons etc.), eine ungewohnte Oberfläche haben oder den Eindruck eines Kontrollinstrumentes erwecken, werden es sicherlich schwerer haben, ihren Zweck zu erfüllen.

- Mangelnde Qualifikation: Nach einer generellen Einführung in ein Programm, sollte der Benutzer durch seine tägliche Arbeit und die Online-Hilfe die entsprechende Qualifikation eigenständig erreichen können (Grundkenntnisse der Anwendungsumgebung und Fachkenntnisse vorausgesetzt).

4.6. Leistungsmerkmale

4.6.1. Datenverwaltung

Zu Beginn der elektronischen Datenverarbeitung standen bei der Anwendungsentwicklung individuelle Programme mit ihren proprietären Ansätzen der Datenspeicherung im Vordergrund. In der Praxis führte das zu Problemen bezüglich

- der Redundanz von Daten: Verschiedene Programme legten Datenstämme mit mehr oder minder großen Überschneidungen an.

- der Aktualität der Daten: Es konnte kaum sichergestellt werden an welcher Stelle Daten gepflegt wurden und welche Änderungen die maßgeblichen waren.

- der Änderung von Programmen: Der Programmier- und Dokumentieraufwand für Änderungen an den Programmen war überdurchschnittlich hoch.

Die Zunahme der Bedeutung von leistungsfähigen Datenverwaltungssystemen führte dazu, daß Daten heute als eigenständiger Teil der betrieblichen DV-Landschaft betrachtet werden. Das Datenaufkommen einer CAS-Software muß also in die bestehenden Strukturen eines Unternehmens passen oder integriert werden können.

Eine Analyse der Datenverwaltung eines Unternehmens ergibt:

- Wo werden von wem welche Daten erzeugt? Diese Primärdaten müssen unter Umständen in die Überlegungen mit einbezogen werden.

- Aus den Primärdaten ableitbare Ergebnisse (Statistiken, Listen, Auswertungen etc.) führen zu dem momentanen Informationsstand der jeweiligen Abteilungen. Hieraus läßt sich der nicht gedeckte Informationsbedarf ersehen.

Die zur Verfügung stehenden Daten werden mit den benötigten Daten abgeglichen. Der zukünftige Datenbedarf wird mit den vorgesehenen Daten der CAS-Software verglichen.

Checkliste:

- Kunden-Stammdaten

- Kunden-Statistikdaten

- Interessentendaten

- Artikeldaten

- Mitarbeiterdaten

- Auftragsdaten

- Angebotsdaten

- Daten über Kundenkontakte

4.6.2. Akquisitionsunterstützung

Die Akquisitionsunterstützung ist Teil der Verkaufsdisposition. Art, Anzahl und Qualität der Kundenkontakte bestimmen die Marktpräsenz der Unternehmen.

Marktpräsenz kann sich äußern in

- persönlichen Kontakten: Besuche beim Kunden, Einladungen der Kunden ins Unternehmen, Zusammentreffen auf Messen, Ausstellungen etc.
- telefonischen Kontakten: Passive Kundenbetreuung (zum Beispiel telefonischer Auftragsannahme) oder aktive Kundenbetreuung (zum Beispiel Telefonverkauf)
- schriftlichen Kontakten: Versand von Prospekten, Direktwerbung, Versenden von Verkaufsförderungsmaterial und Angeboten
- werblichen Kontakten: Anzeigenwerbung, Coupon-Anzeigen, Rundfunkwerbung, Internet etc.

Kosten und Wirksamkeit der einzelnen Kontaktformen sind sehr unterschiedlich. Normalerweise wird der persönliche Kontakt der wirksamste sein, ist aber gleichzeitig auch der teuerste Kontakt. Ihm ist also durch ein CAS-System eine besondere Aufmerksamkeit zu widmen.

Laptop und Datenfernübertragung führen zu besseren Möglichkeiten der Kommunikation zwischen Außen- und Innendienst. Das Zusammenspiel hat sich "vom Papier auf den Draht" verlagert.

Besuchsplanung und –vorbereitung:

In einem zentralen Rechner stehen die aktuellen Daten, die sich aus den Eingaben des Vertriebsinnendienstes und den gesammelten Informationen der Außendienstmitarbeiter zusammensetzen, zur Verfügung. Sie sollen einen tagesaktuellen Überblick über die Kundensituation ermöglichen. Es muß ebenso Möglichkeiten zur Einplanung von kurzfristigen Kundenbesuchen, Wochenplanungen und langfristigen Planungen geben, wobei eine rationelle Tourenplanung eine große Unterstützung darstellt. Umfangreiche Informationen für Gespräche mit Kunden, Neukunden und Interesenten müssen bereitgestellt werden können. Bei einer Verbindung mit der Zentrale sollte der Außendienstmitarbeiter auf Daten über Umsätze, Aufträge, Projekte, Artikel, Lagerbestände, Reklamationstatistiken etc. zugreifen können.

Besuchsdurchführung:

Beim Einsatz eines mobilen Computers bei Verkaufsgesprächen dienen die gewohnten Anwendungsprogramme zur sicheren Bereitstellung und Erfassung von relevanten Informationen. Gibt es die Möglichkeit einer Datenfernübertragung kann dadurch vor Ort eine Angebotsabgabe oder

Auftragserfassung erfolgen, gestützt auf zum Beispiel aktuelle Lagerbestände. Nach wie vor bleibt ein persönliches Verkaufsgespräch mit möglichst wenigen Ablenkungen natürlich der wichtigste Erfolgsgarant. Der Einsatz eines Computers kann lediglich die Aktualität und Qualität von Verkaufsauskünften günstig beeinflußen und soll "doppelte" Arbeit vermeiden.

Besuchsnachbearbeitung und Überwachung:

Nach dem Verkaufsgespräch kann der Computer bei der Erfassung des Besuchsberichtes und der Besuchsnotizen helfen, und es können erste Analysen über Fortschritte oder Mißerfolge durchgeführt werden. Unmittelbar nach einem Verkaufsgespräch kann man die Resultate wesentlich müheloser erfassen, als zum Beispiel später in seinem Büro.

Checkliste:

- Terminplanung (pro Tag, Woche, Monat, Jahr)
- Terminüberwachung (Warnmeldung)
- Terminwiedervorlage (automatische Benachrichtigung)
- Terminvergabe "im Auftrag"
- Sytemgenerierte Terminvorschläge (bezüglich Umsatz, Frequenz etc.)
- Überwachung von Terminüberschneidungen
- Ziel- und Etappenplanung für Kundenkontakte
- Planung der Gesprächsthemen für Kundenkontakte
- Tourenplanung (nach Datum und Zeit, Fahrstreckenoptimierung)
- Informationsbereitstellung über Kunden (flexible Selektionskriterien)
- Berichtserstellung über Kundenkontakte (automatische Wochenberichte)
- Angebotserstellung vor Ort
- Auftragsannahme vor Ort
- Projektsteuerung

4.6.3. Vertriebsplanung, -steuerung und –kontrolle

Das Zusammenspiel zwischen zentralen Daten der Verkaufsabrechnung und dezentralen Daten aus Markt und Kundenumfeld (Verkaufsdisposition) sollen optimal miteinander verzahnt werden und schnell an die richtigen Entscheidungsträger transportiert werden.

Strategische und langfristige Vertriebspläne werden zwar nach wie vor in der Zentrale erstellt und verabschiedet, sie sollen aber wesentlich umfassender auf die Auswertungen der Besuchsberichte zurückgreifen können. Kurzfristige Vertriebspläne können stärker dezentral organisiert werden und mit den betroffenen Außendienstmitarbeitern gestaltet werden.

Informationen, Vorgaben und Kontrolldaten erstrecken sich auf alle Gebiete der Vertriebsplanung:

- Absatz- und Umsatzzahlen (Kunden, Artikel, Gebiete, Deckungsbeiträge)
- Kundenbearbeitung nach Potential, Ausschöpfung etc.
- Zielvorgaben nach Potentialindikatoren (exakte Erfassung der Termine und Kosten)
- Überblick über Angebote, Projekte und Aufträge
- Überprüfung der Wirksamkeit eigener Aktionen

Checkliste:

- Auswertungen über Kunden (Umsätze, Konditionen etc.)
- Auswertungen über Artikel (Artikelgruppen, Reklamationen etc.)
- Auswertungen nach Gebieten (Potentiale, Bevölkerung, Fläche etc.)
- Wettbewerberdaten (Produkte, Gebiete, Umsätze, Marktanteile, Mitarbeiter etc.)
- Auswertungen für die Vertriebsleitung:
- Anzahl der Besuche pro Kunde
- Durchschnittlicher Umsatz pro Besuch
- Deckungsbeitrag pro Außendienstmitarbeiter, Kunde, Artikel etc.
- Anzahl der Neukunden und verlorenen Kunden
- Potentialausschöpfung

- Etc.

4.6.4. Kommunikationsfähigkeit

Bei der traditionellen Form des Betriebsberichtswesens kann man gravierende Nachteile feststellen:

- Es wird viel Papier eingesetzt, um die Kommunikation zwischen Innen- und Außendienst zu gewährleisten. Daten werden wiederholt erfaßt. Es entsteht zusätzlicher Zeitaufwand, und darüber hinaus erhöht sich die Gefahr von Fehlerfassungen und Inkonsistenzen.
- Manche Informationen gehen verloren oder bleiben innerhalb einer Abteilung verschlossen.
- Führungsinstrumente (globale Auswertungen) stehen in angemessenem Umfang nicht im Tagesgeschäft zur Verfügung.
- Die Einarbeitungszeit für neue Außendienstmitarbeiter ist hoch. Die Unterlagen über Kunden und Gebiete sind schwer zugänglich.

Dem gegenüber bietet die elektronische Weitergabe der Daten aus Besuchsberichten Vorteile für die aktuelle Information und schnelle Abstimmung aller Abteilungen eines Unternehmens.

Es sind verschiedene Stufen für die praktische Durchführung denkbar:

- Die Besuchsberichte der Außendienstmitarbeiter werden vom Innendienst erfaßt und ausgewertet.
- Der Außendienstmitarbeiter gibt die Besuchsberichte direkt in einen mobilen Computer vor Ort ein. Der Datenaustausch mit der Zentrale geschieht zum Beispiel über Disketten.
- Die Datenübertragung geschieht per Telefonleitung oder Mobiltelefon über einen Dienstanbieter oder DFÜ (Datenfernübertragung). Auf die Datenbestände kann "online" zugegriffen werden.

Da die Außendienstmitarbeiter aber in keinem Fall ständig mit der Zentrale verbunden sind, muß eine Möglichkeit bestehen, die dezentral gehaltenen Daten mit den zentralen Daten abgleichen zu können.

Checkliste:

- Datenübertragung vom Außendienst zum Innendienst (Besuchsbericht-, Termin-, Angebots-, Auftragsdaten)

- Datenübertragung vom Innendienst zum Außendienst (Stammdaten, Termine)
- Datenübertragung auch mittels Ausdrucken (Besuchsberichte, Termine, Angebote, Aufträge)
- Reaktionsmöglichkeiten auf zukünftige Entwicklungen

4.6.5. Schnittstellen

CAS-Systeme stellen nur einen kleinen Ausschnitt der DV-Landschaft dar. Sie sollten daher Möglichkeiten zur Integration in bestehende und zukünftige Anwendungssysteme bieten. Diese Integration wird meistens dadurch erreicht, daß man definierte Schnittstellen anderer Anwendungen nutzt oder aber eigene Schnittstellen bereitstellt, auf die man von anderen Programmen aus zugreifen kann.

Denkbar sind für eine CAS-Software zum Beispiel Schnittstellen zu:

- Textverarbeitungen
- Tabellenkalkulationen
- Präsentationsprogramme
- Reisekostenabrechnungen
- Projektmanagement-Programme
- Telefonmarketing-Programme
- Auftragsbearbeitungs-Programme
- Fakturierungs-Programme
- Provisionsabrechnungs-Programme
- Email-Systeme

Außerdem sollten Standardformate (ASCII, dbase etc.) sowohl für den Datenimport als auch für den Datenexport unterstützt werden.

Checkliste:

- Schnittstellen zu Standardanwendungen

- Im- und Export von Standardformaten

4.6.6. Bedienung und Softwareergonomie

Da Standardsoftware oft in völlig unterschiedlichen Unternehmen für unterschiedliche Aufgaben eingesetzt wird, muß sie an verschiedene Bedingungen und Bedürfnisse angepaßt werden können.

Man kann dies zum Beispiel durch einen modularen Aufbau der Software erreichen. Die Software wird dann aus den Modulen so zusammengestellt, daß sie den Anforderungen des Unternehmens gerecht wird. Bei, sogar Preis- und Kostenvorteilen, kann teilweise die Individualität einer Individualprogrammierung entstehen. Da aber der Pflege- und Wartungsaufwand erheblich steigt, ist auf lange Sicht bei dieser Alternative mit höheren Kosten zu rechnen.

Ein anderer Ansatz ist es, die Anpassungsmöglichkeiten in dem Programm selber so umfangreich zu gestalten, daß der Anwender oder ein Administrator alle Einstellungen im "laufenden Programm" oder über Werkzeuge vornehmen kann.

Davon abgesehen bleibt eine Software natürlich immer wertlos, wenn sie vom Anwender nicht akzeptiert wird oder nicht bedient werden kann. Die wichtigsten Voraussetzungen auf der Anwenderseite sind zudem Qualifikation und Motivation, die aber auch durch die Software gefördert werden können (Online-Hilfe, "freundliche" Programmgestaltung etc.).

Checkliste:

- Maussteuerung, Funktionstasten und Tastenkombinationen
- Aktive Bedienerführung
- Plausibilitätsprüfung der Eingabe
- Fehlermeldungen im Klartext
- Funktionsmenüs (Leistenmenüs, Pop-Up-Menüs etc.)
- Online-Handbuch, Online-Hilfe
- Makrosprache
- Anpaßbare Menüs und Dialoge
- Sortierung und Suche nach frei wählbaren Kriterien

4.6.7. Datensicherheit

Durch den dezentralen Einsatz des PCs ergeben sich Sicherheitsprobleme hinsichtlich des Datenverlustes, des Datenmißbrauchs und des Datenschutzes. Da räumliche Zugangskontrollen schwer oder gar nicht möglich sind, müssen Vorkehrungen direkt am Computer oder an der Software einsetzen. Außer der Verpflichtung der Benutzer auf das Datengeheimnis (§5 BDSG) müssen noch weitere technische und organisatorische Maßnahmen getroffen werden.

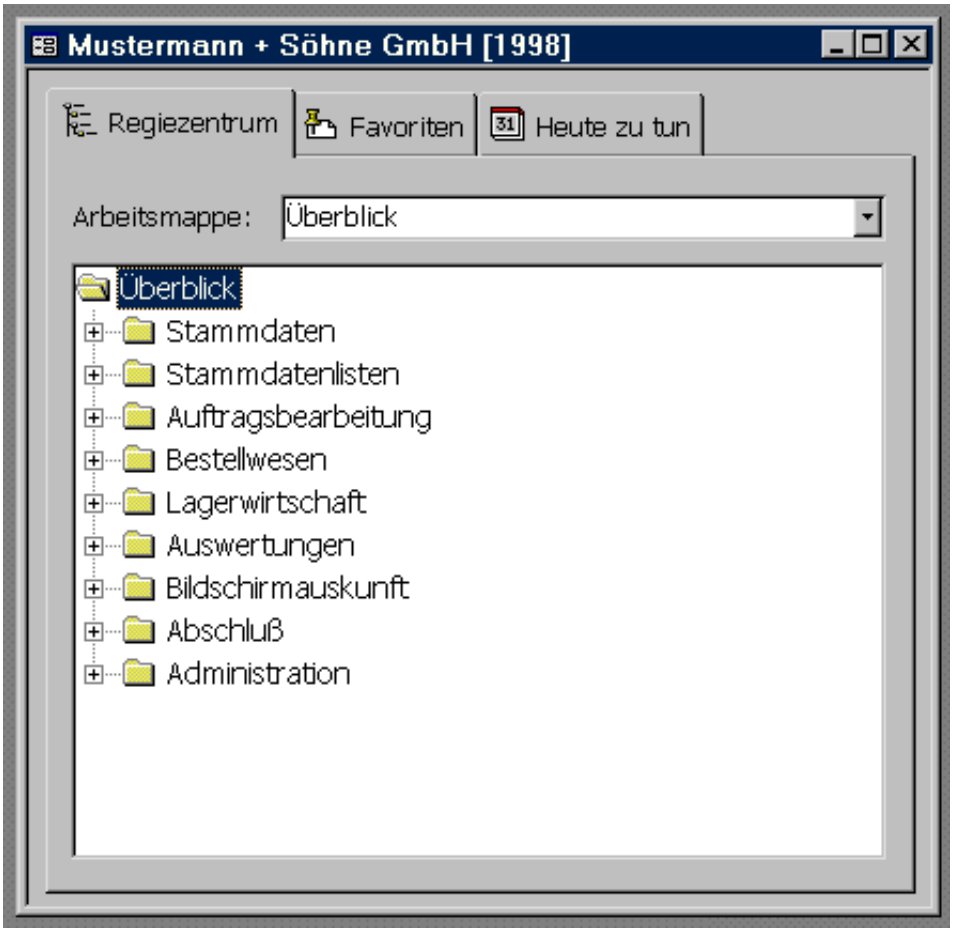
Checkliste:

- Paßwortschutz der Software
- Paßwortänderung durch den Anwender möglich
- Automatischer Vorschlag zur Paßwortänderung (zeitabhängig)
- Zugriffsoperationen (lesen, schreiben etc.) sind einschränkbar
- Meldungen bei Datenänderung und Löschen
- Kopierschutz der Daten
- Datenverschlüsselung für DFÜ und gespeicherte Daten
- Datensicherung im Programm auslösbar
- Wiederherstellungs-Funktionen

Es wird untersucht, welche benötigten Funktionen und Daten schon in der "Sage KHK Office Line Auftragsbearbeitung" vorhanden sind. Die Differenz zu den Anforderungen aus dem vorherigen Kapitel ergibt den notwendigen Funktions- und Datenumfang für das Software Modul "VIS SQL 1.01".

5. Untersuchung der vorhandenen Funktionen und der gegebenen Datenstruktur in der "Sage KHK Office Line Auftragsbearbeitung"

5.1. Programmbeschreibung



Die "Sage KHK Office Line Auftragsbearbeitung" basiert wie die anderen Module der "Sage KHK Office Line" auf einer "Microsoft Access 97" Datenbank. Die globale Datenhaltung ist über ODBC auf einer "Sybase" oder "Microsoft SQL" Datenbank möglich.

Für den Anwender erscheint das Programm als normales Windows-Programm, das über angepaßte Menüleisten und Client-Fenster, die über das sogenannte Regiezentrum gestartet werden, verfügt (Abbildung 5-1). Über die Baumstruktur des Regiezentrums erreicht man die gesamte Funktionalität des Programms. Die Eingabemasken beruhen auf den integrierten Formularen der "Microsoft Access" Datenbank, Auswertungen beruhen meistens auf integrierten Berichten, die vielfältige Möglichkeiten zur Formatierung und zum Ausdruck bieten. Zusätzliche Funktionalität wird durch externe Programmbibliotheken (DLL’s) zur Verfügung gestellt.

Auf das Thema "Microsoft Access 97" wird zu einem späteren Zeitpunkt genauer eingegangen.

5.2. Checkliste Datenverwaltung

- Kunden-Stammdaten:

Die Kunden-Stammdaten werden hauptsächlich in den Tabellen "KHKAdressen" und "KHKAnsprechpartner" verwaltet. Auf die Daten greift man mittels eines Formulars zu, daß man vom Regiezentrum (Abbildung 5-1) aus in dem Ordner "Stammdaten" findet.

- Kunden-Statistikdaten:

Die Kunden-Statistikdaten ergeben sich aus verschiedenen Auswertungen, die man ebenfalls über das Regiezentrum erreichen kann. In dem Ordner "Auswertungen" hat man die Möglichkeit Berichte zu Umsätzen, Lagerbewegungen, Lagerbeständen und Vorgängen zu erzeugen. In dem Ordner "Bildschirmauskunft" findet man eine "Chefübersicht", mit der man sich am Bildschirm grafische Auswertungen anzeigen lassen kann. Außerdem gibt es dort die Möglichkeit zum Zugriff auf Daten

des Verkaufs, des Einkaufs und des Rechnungswesens.

- Interessentendaten:

Wie bei den Kunden-Stammdaten.

- Artikeldaten:

Die Artikeldaten werden hauptsächlich über die Tabelle "KHKArtikel" verwaltet. Auf die Daten greift man mittels eines Formulars zu, daß man vom Regiezentrum (Abbildung 5-1) aus in dem Ordner "Stammdaten" findet.

- Mitarbeiterdaten:

Wie bei den Kunden-Stammdaten.

- Auftragsdaten:

Die Auftragsdaten werden hauptsächlich in den Tabellen "KHKVKBelege" und "KHKVKBelegePositionen" verwaltet. Auf die Daten greift man mittels eines Formulars zu, daß man vom Regiezentrum (Abbildung 5-1) aus in dem Ordner "Auftragsbearbeitung" findet.

- Angebotsdaten:

Wie bei den Auftragsdaten.

- Daten über Kundenkontakte

5.3. Checkliste Akquisitionsunterstützung

- Terminplanung (pro Tag, Woche, Monat, Jahr)
- Terminüberwachung (Warnmeldung)
- Terminwiedervorlage (automatische Benachrichtigung):

Die Auftragsbearbeitung bietet die Möglichkeit Formulare oder Datensätze in ein Wiedervorlagesystem aufzunehmen. Sie erscheinen dann einmalig oder in einem ausgewählten Rhythmus im Register "Heute zu tun" des Regiezentrums (Abbildung 5-1). Außerdem kann ein Status "offen", "zukünftig" oder "erledigt vergeben werden. Da aber ansonsten keine Terminverwaltung existiert, muß diese Funktionalität erweitert werden.

- Terminvergabe "im Auftrag":

Eine Wiedervorlage kann auch für einen anderen, im System bekannten Benutzer generiert werden. Es gelten die gleichen Vorbehalte wie bei der Terminwiedervorlage.

- Sytemgenerierte Terminvorschläge (bezüglich Umsatz, Frequenz etc.)
- Überwachung von Terminüberschneidungen
- Ziel- und Etappenplanung für Kundenkontakte
- Planung der Gesprächsthemen für Kundenkontakte
- Tourenplanung (nach Datum und Zeit, Fahrstreckenoptimierung)
- Informationsbereitstellung über Kunden (flexible Selektionskriterien):

Die Informationsbereitstellung über Kunden beschränkt sich weitgehend auf verschiedene Auswertungsmöglichkeiten bezüglich der Angebots- und Auftragsdaten.

- Berichtserstellung über Kundenkontakte (automatische Wochenberichte)
- Angebotserstellung vor Ort:

Es besteht die Möglichkeit, daß ein Außendienstmitarbeiter mit einer tagesaktuellen Kopie der Ursprungsdatenbank vor Ort arbeitet. Eine Datenreplikation ist aber nicht in der Auftragsbearbeitung vorgesehen und muß noch implementiert werden. Es bieten sich auch Lösungen mit SQL Remote an, welches von Sybase unterstützt wird.

- Auftragsannahme vor Ort:

Wie bei der Angebotserstellung vor Ort.

- Projektsteuerung:

Wie bei der Angebotserstellung vor Ort.

5.4. Checkliste Vertriebsplanung, -steuerung und –kontrolle

- Auswertungen über Kunden (Umsätze, Konditionen etc.):

In der Auftragsbearbeitung stehen vielfältige Möglichkeiten zur Verfügung, Kundendaten über Listen,

Berichte und Diagramme auszuwerten.

- Auswertungen über Artikel (Artikelgruppen, Reklamationen etc.):

Wie bei den Auswertungen über Kunden.

- Auswertungen nach Gebieten (Potentiale, Bevölkerung, Fläche etc.)
- Wettbewerberdaten (Produkte, Gebiete, Umsätze, Marktanteile, Mitarbeiter etc.)
- Auswertungen für die Vertriebsleitung:
- Anzahl der Besuche pro Kunde
- Durchschnittlicher Umsatz pro Besuch
- Deckungsbeitrag pro Außendienstmitarbeiter, Kunde, Artikel etc.
- Anzahl der Neukunden und verlorenen Kunden
- Potentialausschöpfung
- Etc.

5.5. Checkliste Kommunikationsfähigkeit

- Datenübertragung vom Außendienst zum Innendienst (Besuchsbericht-, Termin-, Angebots-, Auftragsdaten)
- Datenübertragung vom Innendienst zum Außendienst (Stammdaten, Termine)
- Datenübertragung auch mittels Ausdrucken (Besuchsberichte, Termine, Angebote, Aufträge):

Ausdrucke sind für Angebote und Aufträge in verschiedenen, variablen Gestaltungen möglich. Termine und Besuchsberichte sind nicht implementiert.

- Reaktionsmöglichkeiten auf zukünftige Entwicklungen:

Die Auftragsbearbeitung ist komplett in "Microsoft Access 97" integriert. So ist man zwar weitgehend von den Entwicklungen, die Microsoft in Zukunft durchführen wird, abhängig, kann aber auf der anderen Seite von der Integration in das "Microsoft Office Paket" profitieren. Mittelfristig ist nicht damit zu rechnen, daß man mit Microsoft-Produkten riskiert, wichtige EDV-Tendenzen zu

versäumen.

5.6. Checkliste Schnittstellen

- Schnittstellen zu Standardanwendungen

In "Microsoft Access 97" hat man vielfältige Möglichkeiten auf Schnittstellen von anderen Programmen zuzugreifen, die sich einerseits durch Windows-Konzepte wie zum Beispiel OLE ergeben, andererseits durch die Nutzung von VBA (Visual Basic for Applications) und ActiveX-Steuerelementen. In der Auftragsbearbeitung wird von diesen Schnittstellen wenig Gebrauch gemacht.

- Im- und Export von Standardformaten

"Microsoft Access 97" kann optional mit Filtern zum Im- und Export der verbreitetsten Datenformate installiert werden. Zur Laufzeit bietet die Auftragsbearbeitung keine Möglichkeiten zum Im- oder Export von Daten.

5.7. Checkliste Bedienung und Softwareergonomie

- Maussteuerung, Funktionstasten und Tastenkombinationen:

Diese Merkmale gehören heute bei einem Programm, daß unter einer grafischen Benutzeroberfläche arbeitet zum Standard.

- Aktive Bedienerführung:

Entsprechend der Aufgabe wird der Benutzer mit Hilfe von Assistenten bei der Problemlösung unterstützt.

- Plausibilitätsprüfung der Eingabe:

Die Plausibilität der Eingabe wird entweder durch die zugrundeliegende Datenbank vor Änderungen geprüft, oder wird teilweise durch Eingabemasken unterstützt.

- Fehlermeldungen im Klartext:

In den meisten Fällen wird der Fehler von der Auftragsbearbeitung aufgefangen, und es wird eine Meldung über die Ursachen angezeigt.

- Funktionsmenüs (Leistenmenüs, Pop-Up-Menüs etc.):

Die Funktionsmenüs sind nicht überladen und stehen für verschiedene Aufgaben zur Verfügung (Pop-Up-Menüs, kontextsensitive Funktionsleisten).

- Online-Handbuch, Online-Hilfe:

An jeder Stelle in der Auftragsbearbeitung kann man mit der Taste "F1" auf die Online-Hilfe (im Zusammenhang) zugreifen.

- Makrosprache:

Eine Makrosprache wird von "Microsoft Access 97" zur Verfügung gestellt und kann in der Auftragsbearbeitung verwendet werden.

- Anpaßbare Menüs und Dialoge

- Sortierung und Suche nach frei wählbaren Kriterien:

Es gibt einen sehr effektiven Suchmechanismus, der einfach auf individuelle Bedürfnisse angepaßt werden kann. Die Sortierung geschieht zum Beispiel mit Hilfe eines sehr flexiblen ActiveX-Steuerelements.

5.8. Checkliste Datensicherheit

- Paßwortschutz der Software:

Jeder Benutzer benötigt ein eigenes Kennwort und ein eigenes Paßwort, um die Auftragsbearbeitung benutzen zu können.

- Paßwortänderung durch den Anwender möglich:

Beim einloggen kann der Benutzer sein aktuelles Paßwort ändern.

- Automatischer Vorschlag zur Paßwortänderung (zeitabhängig)

- Zugriffsoperationen (lesen, schreiben etc.) sind einschränkbar:

Die Benutzer können in verschiedene Gruppen eingeteilt werden und erhalten je nach Gruppenzugehörigkeit individuelle Rechte für die Arbeit in der Auftragsbearbeitung.

- Meldungen bei Datenänderung und Löschen:

Datenänderungen, oder der Versuch Daten zu löschen, werden optisch signalisiert, oder müssen in einem Dialog quittiert werden.

- Kopierschutz der Daten:

Die zugrundeliegende Datenbank ist durch eine "Engine" geschützt und kann weder unbefugt gelesen, noch kopiert oder gelöscht werden.

- Datenverschlüsselung für DFÜ und gespeicherte Daten
- Datensicherung im Programm auslösbar
- Wiederherstellungs-Funktionen

Mit den Ergebnissen aus den Kapiteln 4 und 5 wird eine erste Systemanalyse durchgeführt, die in diesem ersten Schritt als wichtigstes Ergebnis ein Pflichtenheft liefert. Die übrigen Punkte der Systemanalyse fließen in den folgenden Kapiteln weiter ein.

6. Festlegung des Funktions- und Datenumfangs

6.1. Die Systemanalyse

Bei der Produktdefinition werden die Anforderungen an ein Softwareprodukt festgelegt. Das systematische Vorgehen bezeichnet man als Systemanalyse, die folgende Aktivitäten beinhaltet:

a. Anforderungen ermitteln
b. Anforderungen festlegen und beschreiben
c. Anforderungen analysieren
d. Anforderungen simulieren
e. Anforderungen verabschieden

Das Bedürfnis eine neue Software zu produzieren, kann vielfältige Ursachen haben. Es kann durch Innovationen in der EDV-Welt entstehen, auf die man reagieren muß, es kann aber auch durch Unzufriedenheit mit bestehenden Arbeitsabläufen oder durch Produktideen verschiedenster Personengruppen entstehen. Die Anforderungen werden folglich in der ersten Phase immer verschwommen und unvollständig sein.

Auf der anderen Seite ist ein vollständiges und konsistentes Anforderungsdokument nicht nur für den erfolgreichen Verlauf der Softare-Entwicklung wichtig, es dient auch als Basis für die Abnahme des fertigen Produktes.

a. Anforderungen ermitteln

Die Anforderungen an eine Software ergeben sich in erster Linie durch Gespräche mit Personen, die mit dem Programm später arbeiten sollen, oder die nötige Fachkompetenz zur Beurteilung des Leistungsumfangs besitzen. Es kann sich hierbei zum Beispiel direkt um den Auftraggeber, Personen einer Fachabteilung oder sonstige Spezialisten handeln.

Als zweite wichtige Quelle zur Anforderungsbestimmung kann oft auch Fachliteratur zu dem betreffenden Thema oder eine Analyse von Konkurrenzprodukten herangezogen werden. Veröffentlichungen lassen sich heute relativ leicht mit Hilfe der neuen Medien finden und auswerten.

Die Akzeptanz für eine Anwendung kann man schon während der Entwicklung erheblich positiv beeinflussen, wenn man den Endanwender mit einbezieht. Es bieten sich dazu interessante Möglichkeiten, wie zum Beispiel das Internet an. Man kann schon in dieser ersten Phase entworfene Bildschirmmasken und Kurzbeschreibungen zugänglich machen und auf gleichem Wege Resonanz (Emails, Formulare) erhalten. Noch bevor eine Version ausgeliefert wird, kann man so auf Wünsche und Verbesserungsvorschläge eingehen.

Ein wichtiger Punkt kann gegebenenfalls auch die Ist-Analyse der zu ersetzenden oder zu ergänzenden Software sein. Die Mängel, die sich bei einer Ist-Analyse ergeben, führen dann zu neuen Anforderungen, die in die Überlegungen einzubeziehen sind.

Alle Anforderungen liegen nach dieser ersten Phase meistens weder strukturiert noch vollständig vor. Es handelt sich überwiegend um Gesprächsnotizen, vage Ideen und umfassende Unterlagen. Die Notwendigkeit ein konsistentes Produktmodell zu entwickeln ist nun die wichtigste Aufgabe für den Erfolg des Projektes.

- Gespräche mit der Geschäftsleitung und Vertriebsmitarbeitern der "Ihr Partner GmbH"
- Gespräche mit interessierten Kunden der "Ihr Partner GmbH"
- Publikation des TÜV Rheinland "Marktspiegel: CAS-Standardsoftware zur dezentralen Vertriebs- und Außendienststeuerung mit PC"

- Auswertungen über Konkurrenzprodukte
- Internet-Seite mit Bildschirmmasken, Informationen und einem Antwortformular
- Ist-Analyse der Auftragsbearbeitung der "Sage KHK Office Line" (siehe Kapitel 5)

a. Anforderungen festlegen und beschreiben

Nachdem die Anforderungen ermittelt sind, müssen sie festgelegt und beschrieben werden. Man gruppiert und klassifiziert sie zum Beispiel in notwendige und wünschenswerte Anforderungen. Funktionen, Daten, Leistungen und Schnittstellen müssen festgelegt werden.

- Siehe "6.2.2. Pflichtenheft Vertriebsinformationssystem Version 1.01"

a. Anforderungen analysieren

Die beschriebenen Anforderungen sind auf verschiedene Qualitätsziele hin zu analysieren:

- Sind die Anforderungen inhaltlich vollständig?

Vollständigkeit meint den Grad, in dem das Produkt dem Benutzer alle notwendigen Daten und Funktionen zur Verfügung stellt. Wenn hier Diskrepanzen auftreten, ist zu prüfen, ob das Produktziel gefährdet ist, oder ob über Umwege (Zusatzprogramme, Betriebssystem etc.) der gewünschte Einsatz gewährleistet werden kann.

- Sind die Anforderungen konsistent?

Hier wird untersucht, inwieweit die verschiedenen Anforderungen untereinander widerspruchsfrei sind.

- Sind die Anforderungen eindeutig?

Die Anforderungen sollen nur genau eine Interpretation zulassen. Bei verbalen Anforderungen kann es leicht zu Mißverständnissen kommen.

- Sind die Anforderungen durchführbar?

Die Durchführbarkeit ist von verschiedenen Rahmenbedingungen abhängig. Zum Beispiel kann eine bestimmte Antwortzeit des Systems notwendig sein, die aber technisch nicht zu erreichen ist.

- Die Qualitätsziele werden regelmäßig überprüft, da die Kunden in dem Marktsegment, für welches das Vertriebsinformationssystem vorgesehen ist, nicht anonym sind. Auf Wünsche der Kunden muß zu jeder Zeit Rücksicht genommen werden, sofern Konsistenz, Eindeutigkeit und Durchführbarkeit nicht beeinträchtigt werden.

a. Anforderungen simulieren

Die Analyse der Anforderungen reicht manchmal noch nicht aus, um eine klare Vorstellung von einem Softwareprodukt zu erhalten. Man bedient sich daher Methoden und Werkzeugen, die es erlauben, das Verhalten eines Programms schon im Vorfeld der Implementierung zu simulieren.

Die Gestaltung der Benutzungsschnittstellen kann hierbei sehr hilfreich sein. Graphische Entwicklungswerkzeuge gestatten eine komfortable Möglichkeit, Bildschirmmasken zu erzeugen, ohne überhaupt eine Zeile Programmcode geschrieben zu haben (Prototyping).

- Für das Prototyping werden Bildschirmmasken verwendet, die als Bilder mit Hotspots in einem Internetbrowser dargestellt werden. Über die Verzweigungen (Hyperlinks) kann man besonders gut den Programmfluß simulieren, und das zugrundeliegende HTML-Dokument eignet sich für beliebige Kommentare. Die so entstandenen Dokumente lassen sich später als Online-Hilfe nutzen.

a. Anforderungen verabschieden

Die Ergebnisse der Definitionstätigkeiten finden sich in einer Produktdefinition wieder, die meistens aus verschiedenen Teilbereichen besteht. Die entstandenen Dokumente dienen oft als juristische Grundlage und sind Basis für den späteren Softwareentwurf.

- Siehe "6.2.2. Pflichtenheft Vertriebsinformationssystem Version 1.01"

6.2. Das Pflichtenheft

Das Ergebnisdokument einer Anforderungsdefinition wird auch als Pflichtenheft bezeichnet. Es enthält alle fachlichen Anforderungen an das Softwareprodukt aus der Sicht des Auftraggebers, wobei es sich besonders um den Funktions-, Daten-, Leistungs- und Qualitätsumfang des Produktes handelt. Es soll darin grundsätzlich immer nur das "Was" behandelt werden und nicht das "Wie". Das Pflichtenheft wird nach einem standardisierten Gliederungsschema aufgebaut, um es gut lesbar und vergleichbar zu machen. Es wird eine detaillierte verbale Form verwendet, und verschiedenen

Abschnitte werden zur Bezugnahme durchnumeriert. Nach der Planungsphase sollte das Pflichtenheft als eines der ersten Dokument erstellt werden und kann später mit Einverständnis der Beteiligten an geänderte Bedingungen angepaßt werden.

6.2.1. Aufbau eines Pflichtenheftes

1. Zielbestimmung

1.1. Mußkriterien

Die hier aufgeführten Kriterien sind für den Einsatz der Software unbedingt notwendig. Sie müssen für das fertige Produkt erfüllt werden.

1.2. Wunschkriterien

Wunschkriterien sollten in die Überlegungen für eine Software einbezogen werden. Sie sollten so gut wie möglich implementiert werden.

1.3. Abgrenzungskriterien

Diese Kriterien sollen verdeutlichen, was das Produkt nicht leisten kann. Sie spielen eine wichtige Rolle, um Mißverständnisse zwischen den Vertragsparteien über den Leistungsumfang zu reduzieren.

2. Produkteinsatz

2.1. Anwendungsbereiche

In diesem Abschnitt wird festgehalten, in welchen grundsätzlichen Bereichen die Software benötigt wird.

2.2. Zielgruppen

Hier werden die Personen oder Personengruppen aufgeführt, für die das Produkt bestimmt ist.

2.3. Betriebsbedingungen

Die Betriebsbedingungen (Physikalische Umgebung, tägliche Betriebszeit etc.) können einen wesentlichen Einfluß auf die Anforderungen an ein System haben.

3. Produktumgebung

3.1. Software

Hier wird angegeben welche Software auf dem Zielsystem zur Verfügung stehen muß, damit das Produkt lauffähig ist.

3.2. Hardware

Im Abschnitt Hardware wird die minimal erforderliche Konfiguration für das Zielsystem beschrieben.

3.3. Orgware

Mit Orgware sind die organisatorisch notwendigen Rahmenbedingungen für den Einsatz einer Software gemeint.

4. Produktfunktionen

Hier werden alle Produktfunktionen aus der Benutzersicht beschrieben. Die technischen Aspekte stehen ausdrücklich im Hintergrund. Die einzelnen Funktionen werden gekennzeichnet, um sich in anderen Dokumenten darauf beziehen zu können.

5. Produktdaten

Bei den Produktdaten sollen alle langfristig zu speichernden Daten, die im direkten Zusammenhang mit der neuen Software stehen aufgeführt werden. Sie werden ebenfalls zur Referenzierung gekennzeichnet.

6. Produktleistungen

Produktleistungen sind Anforderungen an eine Software, die zeit- oder umfangsbezogen sind. Sie werden gesondert erfaßt und gekennzeichnet.

7. Benutzeroberfläche

Hier werden besondere Bedürfnisse an die Benutzeroberfläche formuliert, die nicht selbstverständlich aus den Anforderungen der Ergonomie hervorgehen. Sie können zur Bezugnahme gekennzeichnet werden.

8. Qualitätsbestimmung

Qualitätsbestimmungen richten sich an verschiedene Aspekte eines Produktes. Sie werden je nach Einsatzgebiet verschieden gewichtet.

9. Globale Testfälle

Die hier definierten Testfälle sind für einen späteren Abnahmetest zu verwenden. Sie zielen auf zentrale und übergreifende Funktionalitäten. Sie werden zur Referenzierung gekennzeichnet.

10. Entwicklungsumgebung

Hier wird aufgeführt, welche Werkzeuge zur Entwicklung verwendet werden.

11. Ergänzungen

Unter Ergänzungen können Zusätze zu den vorherigen Punkten oder spezielle Anforderungen eingetragen werden.

6.2.2. Pflichtenheft Vertriebsinformationssytem Version 1.01

Pflichtenheft Vertriebsinformationssystem Version 1.01

Version Autor Datum Status Kommentar

1.01 Thomas Barthels 5/98 Akzeptiert

1. Zielbestimmung

In die bestehende Auftragsbearbeitung der "Sage KHK Office Line SQL" soll ein Modul zur Vertriebsplanung, Akquisitionsunterstützung und Erfolgskontrolle integriert werden.

1.1. Mußkriterien

- Lückenlose Erfassung der Kundenkontakte
- Schnittstelle zu "Microsoft Outlook" für die Terminverwaltung
- Flexible Auswertungsmöglichkeiten über die Vertriebsaktivitäten
- Schnittstellen zu "Microsoft Excel" und "Microsoft Word"
- Im- und Exportfunktionen für Kundendaten
- Integration in die Benutzeroberfläche der Auftragsbearbeitung der "Sage KHK Office Line SQL"

1.2. Wunschkriterien

- Datenübertragung zwischen Außen- und Innendienst

1.3. Abgrenzungskriterien

- Tourenplanung
- Datensicherung im Programm auslösbar

2. Produkteinsatz

Das Produkt soll zur Akquisition neuer Kunden und zur Betreuung vorhandener Kunden dienen. Es ist speziell auf den Einsatz innerhalb des "Microsoft Office 97" Paketes zugeschnitten und unterstützt die dort enthaltenen Funktionalitäten.

2.1. Anwendungsbereiche

Die Software ist für den kommerziellen Einsatz in Vertriebsabteilungen und für Außendienstmitarbeiter einer Firma entworfen.

2.2. Zielgruppen

Das Produkt ist für Anwender der Auftragsbearbeitung der "Sage KHK Office Line" konzipiert, die ein Hilfsmittel zur Vertriebsunterstützung suchen, dabei nicht auf ihre gewohnte Arbeitsumgebung verzichten möchten und die Funktionen des "Microsoft Office 97" Paketes stärker integrieren möchten.

2.3. Betriebsbedingungen

Büroumgebung.

3. Produktumgebung

Die Software ist für den Betrieb auf einem Arbeitsplatzrechner oder einem mobilen Computer vorgesehen.

3.1. Software

Als Betriebssystem kommen "Microsoft Windows 95 / 98 / NT4" in Frage. Zur Ausführung des Programms wird mindestens eine Laufzeitversion von "Microsoft Access 97" benötigt. Außerdem ist eine lokale oder serverseitige Installation des Datenbank-Management-Systems "Sybase SQL Anywhere" erforderlich.

3.2. Hardware

PC Pentium mit mindestens 32 MB Arbeitsspeicher.

3.3. Orgware

Je nach Einsatzgebiet ist eine Netzwerkverbindung zum Datenbankserver oder zum "Microsoft Exchange Server" notwendig.

3.4. Produktschnittstellen

Die Software hat ihre wichtigste Schnittstelle zu den Kundenstammdaten der Auftragsbearbeitung. Auf der anderen Seite ist die Schnittstelle zum "Microsoft Office 97" Paket, um die Daten mit den

hier vorhandenen Möglichkeiten in Verbindung zu bringen.

4. Produktfunktionen

4.1. Verwaltung der Kunden

F10: Die Kunden können mit Kennzeichen versehen werden. Die Werte für die Kennzeichen können gesetzt, geändert oder gelöscht werden.

F20: Die Kennzeichen werden über einen zentralen Menüeintrag und einen speziellen Dialog verwaltet. Wird ein Kennzeichen entfernt, so werden nach einer Warnung auch alle Verweise auf das entsprechende Kennzeichen gelöscht.

F25: Einem Kunden können beliebig viele Kontakte zugeordnet werden. Zu jedem Kunden erscheinen die Kontakte in einer Liste und können bearbeitet oder gelöscht werden.

F30: Die Kunden können durch beliebige Stichworte gekennzeichnet werden. Ein Stichwort kann einem Kunden zugeordnet oder wieder entfernt werden.

F40: Die Stichworte werden über einen zentralen Menüeintrag und einen speziellen Dialog verwaltet. Wird ein Stichwort entfernt, so werden nach einer Warnung auch alle Verweise auf das entsprechende Stichwort gelöscht.

F50: Einem Kunden können beliebig viele Dokumente zugeordnet werden. Die Verweise auf die Dokumente erscheinen in einer Liste und können nach einer Sicherheitsabfrage gelöscht werden.

F60: Ein Doppelklick auf einen Dokumentenverweis öffnet das zugehörige Dokument mit der in Windows registrierten Anwendung.

F70: Der Kunde kann über ein kontextsensitives Menü einer beliebigen Zielgruppe zugeordnet werden.

F80: Für den Kunden kann über ein kontextsensitives Menü ein Standardanschreiben über "Microsoft Word" erzeugt werden. Das Standardanschreiben ist ein Kontakt, und es wird daher automatisch ein neuer Kundenkontakt mit einem Dokumentenverweis auf das Standardanschreiben erzeugt.

F90: Die Standardanschreiben werden über einen zentralen Menüeintrag und einen speziellen Dialog verwaltet. Sie können benannt werden, was sich unmittelbar auf das entsprechende Kontextmenü auswirkt, und über einen Dateidialog lokalisiert werden.

4.2. Verwaltung der Ansprechpartner

F100: Die Ansprechpartner können über einen separaten Dialog angezeigt werden. In einem Suchbrowser können sie nach verschiedenen Kriterien ausgewählt werden. Wählt man einen Ansprechpartner aus, wird in dem Dialog der zugehörige Kunde mit Adresse, Telefon und Telefax angezeigt.

F110: Über den Ansprechpartner-Dialog kann man einen Dialog über die Details zum Ansprechpartner erreichen. Dort findet man außerdem persönliche Daten zum Ansprechpartner, eine Memofunktion und eine Liste der Selektionskriterien für den Ansprechpartner.

F120: Einem Ansprechpartner können beliebig viele Selektionskriterien zugeordnet oder wieder entfernt werden.

F130: Die Selektionskriterien werden über einen zentralen Menüeintrag und einen speziellen Dialog verwaltet. Wird ein Selektionskriterium entfernt, so werden nach einer Warnung auch alle Verweise auf das entsprechende Selektionskriterium gelöscht.

F140: Über den Detaildialog zum Ansprechpartner kann man dessen Briefadresse in die Windows-Zwischenablage kopieren, ihn als Kontakt (in Outlook wird eine Adresse als Kontakt bezeichnet) nach "Microsoft Outlook" exportieren, ihm eine Email schicken oder ein Anschreiben an ihn mit "Microsoft Word" erzeugen.

F150: Bei einer Email oder einem Anschreiben an den Ansprechpartner wird automatisch ein Kontakt mit einem Verweis auf das jeweilige Dokument erzeugt.

4.3. Verwaltung der Kundenkontakte

F160: Die Kundenkontakte werden zu jedem Kunden über einen einheitlichen Dialog erfaßt. Standardmäßig wird ein neuer Kontakt mit der aktuellen Uhrzeit und dem aktuellen Datum versehen. Zu jedem Kontakt hat man die Möglichkeit ein Memo und beliebig viele Verweise auf Dokumente zu hinterlegen oder wieder zu entfernen.

F170: Die Dokumentenverweise werden in einer Liste dargestellt, und man öffnet mit einem Doppelklick das zugehörige Dokument über das dafür registrierte Windows-Programm.

F180: Kundenkontakte können in das Wiedervorlagesystem der Auftragsbearbeitung der "Sage KHK Office Line" übernommen werden.

F190: Ein Kundenkontakt kann dupliziert werden. Der daraus entstandene neue Kundenkontakt unterscheidet sich lediglich dadurch, daß er die aktuelle Uhrzeit und das aktuelle Datum enthält.

F200: Jeder Kundenkontakt kann als Aufgabe nach "Microsoft Outlook" exportiert werden. Es wird vor dem Export eine Fälligkeit für die Aufgabe abgefragt.

F210: Die Kontakte werden für jeden einzelnen Kunden in einer Liste dargestellt. Von dort aus kann man jeden Kontakt über den Erfassungsdialog bearbeiten.

F220: Die Kundenkontakte können über eine globale Liste bearbeitet werden. Als Standardeinstellung werden alle Kundenkontakte des aktuellen Benutzers angezeigt. Die Kundenkontakte können nach beliebigen Kriterien sortiert und gefiltert werden.

F230: In der globalen Liste kann man über ein kontextsensitives Menü Kundenkontakte nach einer Sicherheitsabfrage löschen, zum zugehörigen Kunden wechseln, oder den Kontakt über den Erfassungsdialog bearbeiten.

4.4. Verwaltung von Aktionen

F240: Aktionen werden über einen eigenständigen Dialog verwaltet. Über einen Suchbrowser kann man die Aktionen nach verschiedenen Kriterien auflisten.

F250: Wählt man eine Aktion aus, so werden die gegebenenfalls zugehörigen Kontakte aufgelistet.

F260: Die aufgelisteten Kontakte können über ein kontextsensitives Menü bearbeitet oder gelöscht werden. Man kann über dieses Menü zum entsprechenden Kunden wechseln.

F270: Eine Aktion kann zusammen mit den gegebenenfalls zugehörigen Kontakten im Aktionen-Dialog nach einer Sicherheitsabfrage gelöscht werden.

4.5. Verwaltung von Projekten

F280: Projekte werden über einen eigenständigen Dialog verwaltet. Über einen Suchbrowser kann man die Projekte nach verschiedenen Kriterien auflisten.

F290: In dem Dialog kann ein neues Projekt angelegt werden, ein Projekt gelöscht werden und die Kontakte zu einem Projekt angezeigt werden.

F300: Wird ein neues Projekt angelegt, kann man den Beginn- und Endtermin dafür festlegen. Je nach Bearbeitungsstatus wird visuell auf den Gesamtstatus des Projektes aufmerksam gemacht.

F310: Löscht man ein Projekt, so kann man nach einer Warnmeldung entweder nur das Projekt oder auch alle zugehörigen Kontakte löschen.

F320: Die Kontaktliste zu einem Projekt erlaubt das Löschen von Kontakten, die Bearbeitung von Kontakten und die Anzeige des jeweils zugehörigen Kunden.

4.6. Verwaltung von Zielgruppen

F330: Die Zielgruppen werden über einen eigenständigen Dialog verwaltet. Über einen Suchbrowser kann man die Zielgruppen nach verschiedenen Kriterien auflisten.

F340: In dem Dialog kann eine neue Zielgruppe angelegt werden oder eine Zielgruppe nach einer Sicherheitsabfrage gelöscht werden.

F350: Für eine Zielgruppe können beliebig viele Bedingungen erzeugt werden. Sie werden über umgangssprachliche Eingabemöglichkeiten bestimmt und in einer Liste als SQL-Bedingung angezeigt.

F360: Die SQL-Bedingungen können hinzugefügt, gelöscht und bearbeitet werden.

F370: Mit den Zielgruppen können verschiedene Aktionen durchgeführt werden. Es kann eine Vorschau der Zielgruppe angezeigt werden, es kann ein Serienbrief mit "Microsoft Word" erzeugt werden, es kann eine Serien-Email über "Microsoft Outlook" verschickt werden, die Zielgruppe kann nach "Microsoft Excel" exportiert werden, die Zielgruppe kann mit einem Kennzeichen versehen werden, oder es kann ein Sammelkontakt für die Zielgruppe erzeugt werden.

F380: Die Vorschau zeigt in einem Dialog alle Kunden einer Zielgruppe an. Diese können hier für die aktuelle Aktion ausgeschlossen oder dauerhaft aus der Zielgruppe entfernt werden.

F390: Über einen Dialog kann man für den Serienbrief bestimmen, ob ein vorhandenes Word-Dokument verwendet werden soll, oder ob ein neues Word-Dokument erzeugt werden soll. Man kann außerdem bestimmen, ob der Serienbrief automatisch entsprechende Kundenkontakte erzeugt. Es wird eine Aktion ‚Microsoft-Word Serienbrief' mit Datum und Uhrzeit erzeugt.

F400: Das Serien-Email wird über einen Dialog erstellt. Man hat die Möglichkeit, eine Aktionsbezeichnung zu vergeben, eine Betreffzeile einzutragen, automatisch eine Anrede erzeugen zu lassen, Textbausteine aus der Auftragsbearbeitung der "Sage KHK Office Line" in den Textkörper zu übernehmen, Einzel-Emails oder eine Sammel-Email erzeugen zu lassen, Dateien anzuhängen und Kundenkontakte für das Serien-Email zu erzeugen. Die Angabe einer Aktionsbezeichnung ist für die automatisch erzeugte Aktion erforderlich.

F410: Der Export nach "Microsoft Excel" geschieht über einen Dateidialog, wobei man die Möglichkeit hat, einen Verzeichnispfad auszuwählen und einen Dateinamen einzutragen. Es wird eine Aktion ‚Export nach Excel' mit Angabe der zugehörigen Zielgruppe erzeugt.

F420: Die Änderung eines Kennzeichens für eine Zielgruppe wirkt sich auf alle Kunden der Zielgruppe aus. Über einen Dialog wird das Kennzeichen und der Wert bestimmt. Es wird eine Aktion ‚Sammelkennzeichnung' mit Angabe der zugehörigen Zielgruppe und geändertem Kennzeichen erzeugt.

F430: Ein Sammelkontakt wird über den Standard-Kontakt-Dialog erzeugt. Beim Schließen wird nachgefragt, ob der Sammelkontakt gespeichert werden soll. Wird der Sammelkontakt gespeichert, wird eine Aktion ‚Sammelkontakt' mit Datum und Uhrzeit erzeugt.

4.7. Importfunktion

F440: Die Importfunktion für verschiedene Textformate wird über einen eigenständigen Dialog angeboten. Über einen Dateidialog kann man die entsprechende Datei auswählen. Sie wird auf ein erkennbares Format überprüft und kann dann über einen Texteditor geöffnet und bearbeitet werden.

F450: Das Ergebnis wird tabellarisch dargestellt und kann bezüglich des Formats angepaßt werden.

F460: Die Spalten werden zusätzlich in Zeilen eingeteilt. Die einzelnen Felder können separat bearbeitet werden.

F470: Die Spalten werden den Spalten der Bestimmungstabelle zugewiesen.

F480: Nach einer Prüfung der Vollständigkeit der Angaben können die Datensätze übernommen werden.

5. Produktdaten

5.1. Kundendaten

D5: Die Kunden werden um jeweils ein Feld für den zuständigen Innendienstmitarbeiter, den zuständigen Außendienstmitarbeiter und den zuständigen Projektleiter erweitert.

D10: Die Kunden sollen nach beliebig vielen Kennzeichen eingeteilt werden können. Dazu werden frei definierbare Felder angehängt, die mit vorgegebenen Werten belegt werden können.

D20: Um Kunden kurzfristig gruppieren zu können, können Stichworte generiert werden, die auf einen Kunden zutreffen oder nicht. Das Stichwort enthält einen Index, eine Bezeichnung und den Verweis auf einen Kunden.

D30: Einem Kunden können Dokumente zugeordnet werden. Es wird ein Verweis auf die Dokumente mit einem Index und einem Verweis auf den Kontakt abgespeichert.

5.2. Ansprechpartnerdaten

D40: Die Daten zu den Ansprechpartnern sollen um den Vornamen, den Titel, die Position, den Beruf, das Büro, den Vorgesetzten, den Sekretär, die Anrede, die Briefanrede und die persönlichen Daten (Straße, Land, PLZ, Ort, Telefon, Telefax, Mobiltelefon, Email, Partner, Hobbys, Geburtstag, Jahrestag, Spitzname) ergänzt werden, um eine Konformität zu den Daten in "Microsoft Outlook" zu erreichen. Über ein Kennzeichen soll ein Hauptansprechpartner ausgewählt werden können.

D50: Zur Gruppierung der Ansprechpartner sollen beliebige Kriterien definiert werden können. Das Kriterium wird mit Index, Bezeichnung und Verweis auf den Ansprechpartner abgespeichert.

5.3. Daten über Kundenkontakte

D60: Ein Kontakt wird einem Kunden oder Interessenten über dessen Index und unter Berücksichtigung des Mandanten zugeschlüsselt. Der Kontakt enthält einen Index, die Uhrzeit, ein Datum, ein Benutzerkürzel, eine Bezeichnung, einen Text, eine Kontaktart, einen Ansprechpartner, ein Ergebnis, einen Kommentar des Benutzers, einen Kommentar des Kunden und ein Memofeld.

D70: Ist der Kontakt durch eine Aktion entstanden, so ist das in dem Kontakt zu vermerken.

D80: Gehört der Kontakt zu einem Projekt, so ist das in dem Kontakt zu vermerken.

D90: Hinterlegt man zu einem Kontakt Dokumente, dann wird ein Verweis auf diese mit einem Index und einem Verweis auf den Kontakt abgespeichert.

5.4. Daten über Aktionen

D100: Die Aktionen Serienbrief, Serien-Email, Export nach Excel, Sammelkennzeichnung und Sammelkontakt werden mit Index, Bezeichnung, Datum, Benutzer und Mandant gespeichert.

5.5. Daten über Projekte

D110: Projekte werden über einen Index und eine eindeutige Bezeichnung erzeugt. Sie enthalten außerdem den Benutzer, den Mandanten, den Anfangstermin, den Endtermin und einen Status.

5.6. Daten über Zielgruppen

D120: Zielgruppen enthalten einen Index, eine Bezeichnung, den Benutzer, den Mandanten, eine Kategorie, die Gültigkeit, eine SQL-Anweisung und ein Memo-Feld.

D130: Zielgruppen können durch verschiedene Bedingungen ergänzt werden. Die Bedingungen besitzen einen Index und eine SQL-Bedingung und sind der Zielgruppe über einen Schlüssel zugeordnet.

5.7. Einfache Datenspeicher

D140: Ein Datenspeicher für Programmvariablen wird anstelle einer INI-Datei eingesetzt.

D150: Ein Datenspeicher nimmt die definierten Kontaktarten auf.

D160: Ein Datenspeicher nimmt die definierten Kontakthemen auf.

D170: Ein Datenspeicher nimmt die definierten Kontaktergebnisse auf.

6. Produktleistungen

L10: Für die Kundenkontakte (D60) ist mit einem jährlichen Datenaufkommen von ca. 10000 Datensätzen pro Benutzer zu rechnen.

L20: Bei Übersichtsfunktionen ist eine Antwortzeit von 10 Sekunden nicht zu überschreiten.

7. Benutzungsschnittstellen

B10: Die Bedienungsoberfläche ist soweit wie möglich an die Oberfläche der Auftragsbearbeitung der "Sage KHK Office Line" anzupassen.

B20: Für häufig verwendete Funktionen werden Tastenkombinationen zur Verfügung gestellt.

B30: Die Benutzungsoberfläche kann an die Bedürfnisse der verschiedenen Anwender angepaßt werden.

8. Qualitätsbestimmung

Produktqualität: sehr gut gut normal nicht relevant

Funktionalität

Angemessenheit X

Richtigkeit X

Interoperabilität X

Ordnungsmäßigkeit X

Sicherheit X

Zuverlässigkeit

Reife X

Fehlertoleranz X

Wiederherstellbarkeit X

Benutzbarkeit

Verständlichkeit X

Erlernbarkeit X

Bedienbarkeit X

Effizienz

Zeitverhalten X

Verbrauchsverhalten X

Änderbarkeit

Analysierbarkeit X

Modifizierbarkeit X

Stabilität X

Prüfbarkeit X

Übertragbarkeit

Anpaßbarkeit X

Installierbarkeit X

Konformität X

Austauschbarkeit X

---

9. Globale Testfälle

T10: Kontakterfassung, Kontaktänderung, Dokumentzuordnung, Kundenkennzeichnung,

Ansprechpartnerkennzeichnung, Zielgruppendefinition, Projektdefinition, Aktionsübersicht

T20: Sammelkennzeichnung, Sammelkontakte erzeugen

T30: "Microsoft Office 97" Funktionalitäten

T40: Adressenimport

T50: Löschen von Kontakten mit den zugehörigen Dokumentverweisen, Löschen von Aktionen mit den zugehörigen Kontakten, Löschen von Projekten mit den zugehörigen Kontakten

10. Entwicklungsumgebung

"Microsoft Access 97" wird als Entwicklungsumgebung genutzt. Die Programmierung erfolgt mit "Visual Basic for Applications".

Entsprechend den Vorgaben aus dem Pflichtenheft wird an dieser Stelle eine objektorientierte Analyse durchgeführt. Zunächst wird die verwendete Notation beschrieben und dargestellt, danach wird schrittweise ein Klassendiagramm entwickelt. Das vollständige Klassendiagramm befindet sich im Anhang.

7. Die objektorientierte Analyse

7.1. Gründe für eine objektorientierte Vorgehensweise

Softwaresysteme sind in der Regel so groß, daß sie nicht von einzelnen Personen entwickelt werden können. Für einen einzelnen ist es völlig unmöglich, ein solches System als Ganzes zu verstehen. Aus diesem Grund müssen Mittel gefunden werden, die eine Beherrschung der Komplexität möglich machen. Mit dem strukturierten Ansatz wurden zur Entschärfung der Komplexität das Konzept der Abstraktion (prozedural und datenorientiert) sowie das Prinzip der Datenkapselung eingeführt.

Beim objektorientierten Ansatz werden diese Prinzipien weitergeführt, indem mit der Einführung von Objekten Daten, und die auf die Daten einwirkenden Operationen zusammengefaßt werden. Außerdem wird Vererbung und die Kommunikation mit Botschaften eingeführt.

Im klassischen Software-Engineering erfolgt die Modellierung durch Datenfluß- und Entity-Relationship-Diagramme. Diese Darstellungen bieten unterschiedliche Sichtweisen eines Problembereichs, die in keiner festen Beziehung zueinander stehen. Dadurch ist das Modell relativ unflexibel gegenüber Änderungen. Außerdem bilden solche Diagramme nur jeweils einen Teil eines Problembereichs ab. Bei der klassischen Methode sind Analyse und Design zwei streng voneinander getrennte Phasen, in denen auch unterschiedliche Notationen verwendet werden. Diese Eigenschaften behindern eine kontinuierliche Weiterentwicklung des Modells von der Analyse bis zum Design.

Der objektorientierte Ansatz bietet den Vorteil, daß der Problembereich direkt auf das Modell abgebildet wird. Die Übergänge von OOA über OOD zu OOP sind fließend und es wird eine

durchgehend einheitliche Notation gewählt, so daß das erstellte Modell verständlicher und flexibler gegenüber Änderungen ist. Komponenten, die mit objektorientierten Techniken entwickelt wurden, sind leichter wiederverwendbar, da Klassen abgeschlossene Einheiten bilden, die sowohl Daten als auch Operationen definieren.

7.2. Verwendete Strukturen und Notationen

7.2.1. Objekte

Ein Objekt ist der Gegenstand des Interesses. Es kann Grund für Beobachtungen, Untersuchungen oder Messungen sein. Es ist ein individuelles Exemplar von Dingen, Personen oder Begriffen der realen Welt oder der Vorstellungswelt.

Ein Objekt besitzt definierte Eigenschaften und reagiert mit einem beschreibbaren Verhalten auf seine Umgebung. Alle Eigenschaften eines Objektes werden durch dessen Attributwerte (Daten) bestimmt, sein Verhalten wird durch die Menge seiner Operationen beschrieben. Eine wichtige Eigenschaft jedes Objektes ist seine Identität, mit der es von allen anderen Objekten unterschieden werden kann. Besitzen Objekte identische Eigenschaften, so kann man sie immer über ihre Identität gegeneinander abgrenzen.

Die Daten eines Objektes können nur über dessen Operationen gelesen oder verändert werden. Diese Verkapselung der Daten führt dazu, daß sie außerhalb des Objektes nicht sichtbar sind.

Ein Objekt ist folglich vergleichbar mit einer Entität (ER-Modell). Zusätzlich zu einer Entität besitzt es aber noch eigene Operationen.

- Verwendete Notation:

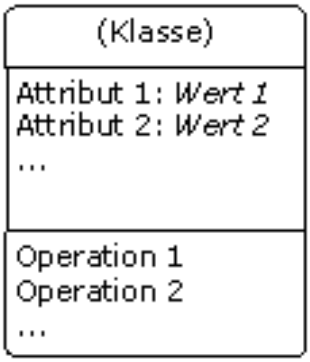


Abbildung 7-1

7.2.2. Klassen

Eine Klasse ist eine Gruppe von Dingen, Lebewesen oder Begriffen mit gemeinsamen Merkmalen.

In objektorientierten Modellen stellt eine Klasse eine Menge von Objekten mit denselben Attributen und Operationen dar. Sie dient gleichzeitig bei der Erzeugung von neuen Objekten als "Schablone" und stellt einen Mechanismus zu deren Erzeugung bereit (konkrete Klasse).

Eine Klasse, von der keine Objekte erzeugt werden können, nennt man dagegen abstrakte Klasse. Eine solche Klasse ist besonders für die Konzepte der Vererbung von Bedeutung.

Man benennt eine Klasse durch ein Substantiv im Singular und gegebenenfalls durch ein zusätzliches Adjektiv. Sie wird grafisch und textuell beschrieben, wobei die grafische Darstellung den Klassennamen, die Attributnamen und die Operationsnamen enthält und durch weitere Informationen in Textform ergänzt werden kann.

- Verwendete Notation:

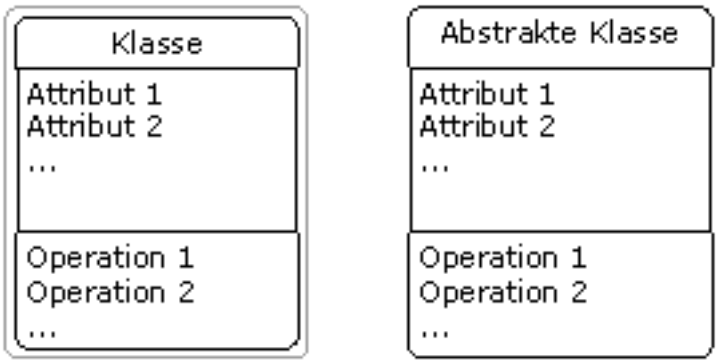


Abbildung 7-2

7.2.3. Attribute

Die Attribute beschreiben die Eigenschaften einer Klasse. Alle Objekte einer Klasse besitzen die gleichen Attribute mit normalerweise unterschiedlichen Werten. Auch wenn Attribute nicht mit Werten belegt sind, existieren sie für das betreffende Objekt (Speicherplatz).

Innerhalb einer Klasse muß die Bezeichnung eines Attributes eindeutig sein. Im allgemeinen wird dafür ein Substantiv verwendet.

Auf die Attributwerte darf nur mit den Operationen der entsprechenden Klasse zugegriffen werden. Sie sind für andere Klassen und deren Objekte nicht sichtbar.

Ein Attribut kann ein Muß- oder ein Kann-Attribut sein. Ein Muß-Attribut muß zu jedem Zeitpunkt der Existenz eines Objektes einen Wert besitzen. Die Initialisierung muß schon bei der Erzeugung des

Objektes geschehen. Ein Kann-Attribut kann seinen Wert jederzeit oder auch gar nicht zugewiesen bekommen.

Schlüsselattribute sind bezüglich ihrer Werte innerhalb einer Klasse eindeutig. Eine Ausnahme besteht nur, wenn in einer Klasse mehrere Attribute gleichzeitig zur Identifikation eines Objektes herangezogen werden. Die Kombination dieser Attributwerte muß dann einen eindeutigen Schlüssel ergeben. Bei der Kombination ist darauf zu achten, daß eine minimal notwendige Anzahl von Attributen als Schlüssel gewählt wird.

Als Werte für Attribute kommen entweder einfache Datenelemente oder definierte Datenstrukturen in Frage. Strukturierte Typen reduzieren die Anzahl der Attribute und verbessern die Lesbarkeit einer Klasse. Zusätzlich können auch elementare Typen selber definiert werden, wenn das betreffende Attribut dadurch problemnäher beschrieben wird.

Einen Sonderfall stellt das sogenannte Klassenattribut da. Es beschreibt eine Eigenschaft einer Klasse, während Attribute Eigenschaften der einzelnen Objekte beschreiben. Ein Klassenattribut liegt vor, wenn für alle Objekte einer Klasse nur ein Wert existiert. Es ist unabhängig von den Objekten und kann auch dann einen Wert besitzen, wenn überhaupt kein Objekt zu einer Klasse erzeugt wurde. Ein Klassenattribut wird durch ein nachgestelltes "(K)" gekennzeichnet.

- Verwendete Notation:

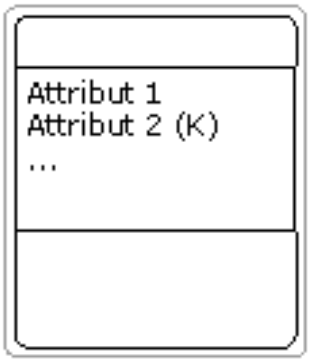


Abbildung 7-3

7.2.4. Operationen

- Objektoperationen:

Sie stellen die Dienstleistungen dar, die ein Objekt seiner Umgebung zur Verfügung stellt und bestimmen somit die Funktionalität des Objektes.

Es handelt sich um ausführbare Tätigkeiten, die mit dem aufrufenden Objekt über Ein- und Ausgabeparameter kommunizieren. Die Semantik wird in einer Spezifikation festgelegt.

Alle Objekte einer Klasse benutzen dieselben Operationen, wobei diese auf ein einzelnes vorhandenes Objekt angewendet werden. Die betreffende Operation kann auf alle Attribute des Objektes zugreifen.

Hilfsoperationen, die nur innerhalb einer Klasse selbst benötigt werden, werden nicht in der Darstellung eines Problems berücksichtigt. Man erkennt sie daran, daß sie keine Leistungen für andere Klassen oder für den Benutzer zur Verfügung stellen.

Man kann Operationen nach ihren Aufgaben klassifizieren:

1. Operationen mit lesendem Zugriff auf die Attribute der eigenen Klasse
2. Operationen mit schreibendem Zugriff auf die Attribute der eigenen Klasse
3. Operationen zur Durchführung von Berechnungen

- Klassenoperationen:

Eine besondere Form der Operationen stellen die Klassenoperationen dar. Sie sind einer Klasse zugeordnet und stehen einem einzelnen Objekt dieser Klasse nicht zur Verfügung. Klassenoperationen werden immer auf mehrere oder alle Objekte einer Klasse angewendet, oder sie manipulieren Klassenattribute. Eine solche Operation wird analog zu den Attributen am Ende des Namens mit einem "(K)" gekennzeichnet.

- Erzeugungsoperationen:

Eine weitere Sonderform der Operationen ist die Erzeugungsoperation. Sie wird dazu benötigt, ein neues Objekt einer Klasse zu erzeugen und deren Muß-Attribute mit entsprechenden Werten zu initialisieren.

Die Erzeugungsoperation gehört zu den impliziten Operationen einer Klasse. Diese Operationen werden von fast jeder Klasse benötigt. Normalerweise gibt es zusätzlich noch Operationen zum löschen von Objekten und zum ändern und lesen von Attributwerten. Diese Operationen werden in einem Klassendiagramm normalerweise nicht dargestellt.

- Operationsbezeichnung:

Der Name einer Operation sollte ausdrücken, was diese leistet. Im allgemeinen wird er also ein Verb enthalten und gegebenenfalls das Attribut, das betroffen ist. Operationen die lediglich den Status eines Attributes ermitteln, können durch den Attributnamen gefolgt von einem Fragezeichen benannt werden. Der Operationsname muß innerhalb einer Klasse eindeutig sein. Außerhalb der Klasse wird zusätzlich zum Operationsnamen der Klassenname vorangestellt (typischerweise Punktnotation).

Außer durch den Namen wird eine Operation noch durch ihre Ein- und Ausgabeparameter beschrieben. Sie enthalten alle Informationen, über die eine Operation mit dem aufrufenden Objekt

kommuniziert.

- Operationsspezifikation:

Die Spezifikation einer Operation beschreibt deren Aufgaben aus fachlicher Sicht. Sie kann Botschaften (siehe unten) an andere Objekte enthalten, die zur Lösung von Teilaufgaben dienen.

Zur Beschreibung wird eine Mischung aus Standardbefehlen und freiem Text verwendet. Formalismus ist nur erforderlich, wenn es um Vollständigkeit und Konsistenz geht, ansonsten soll freier Text das Verständnis verbessern.

- Verwendete Notation:

```
[Klasse.]Operation (in Parameter I1, Parameter I2, ...; out Parameter O1,
Parameter O2, ...)

Freier Text (Pseudocode).
```

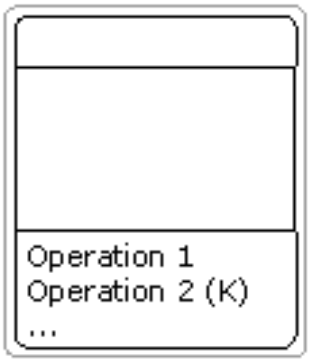


Abbildung 7-4

7.2.5. Vererbungen

Wenn eine Klasse über die Eigenschaften und das Verhalten von einer anderen Klasse verfügen kann, liegt eine Vererbungsstruktur vor. Das geerbte Verhalten einer Klasse kann darüber hinaus redefiniert werden.

Die Klasse, von der eine andere Klasse erbt, bezeichnet man als deren Oberklasse. Eine direkte Oberklasse ist eine Klasse, die sich in einer Klassenhierarchie am nächsten zu einer erbenden Klasse befindet.

Dementsprechend sind alle Klassen, die von einer anderen Klasse erben deren Unterklasse. Eine direkte Unterklasse ist in der Klassenhierarchie direkter Nachbar einer Oberklasse.

Eine Klasse kennt nur die Attribute und Operationen ihrer Oberklassen. Dagegen sind ihr die

Attribute und Operationen ihrer Unterklassen unbekannt.

Man verbindet Oberklasse und Unterklasse mit einer Linie. Durch einen Halbkreis kennzeichnet man die Vererbungsstruktur. Oberklassen sollten bei der grafischen Darstellung über den Unterklassen angeordnet werden.

Bei der Einfachvererbung kann eine Klasse nur eine direkte Oberklasse haben. In jedem Ast einer Baumstruktur kann es folglich nur eine Wurzelklasse geben. Bei der Mehrfachvererbung können auch mehrere direkte Oberklassen zu einer Klasse existieren. Die Baumstruktur kann hier in einem Ast auch mehrere Wurzeln haben.

Für den Fall das bei der Mehrfachvererbung eine Klasse von zwei Oberklassen Attribute oder Operationen mit dem gleichen Namen erbt, muß eine Strategie zur Lösung des Konfliktes entwickelt werden.

Eine abstrakte Klasse kann ihre Operationen entweder nur als Signatur vererben (abstrakte Operationen), oder sie besitzt vollständig spezifizierte Operationen, ohne jedoch Objekte erzeugen zu können.

Enthält eine Unterklasse eine Operation, die in einer Oberklasse mit dem gleichen Namen existiert, redefiniert sie diese Operation. In einer redefinierten Operation müssen die Anzahl und die Typen der Ein- und Ausgabeparameter mit denen der Ursprungsoperation übereinstimmen. Die Schnittstelle der neuen Operation muß also konform zur Operation der Oberklasse sein.

Existieren zu der redefinierten Operation Vor- oder Nachbedingungen in der Oberklasse, so sind diese zu beachten. Vorbedingungen können beibehalten oder gelockert werden, Nachbedingungen können beibehalten oder verschärft werden.

Das Konzept der Vererbung bietet folgende Vorteile:

- Man kann auf existierende Klassen aufbauen, um mit relativ kleinem Aufwand neue Klassen zu erschaffen.

- Die Änderbarkeit von Strukturen wird unterstützt. Eine Änderung in einer Oberklasse wirkt sich auf alle ihre Unterklassen aus.

Auf der anderen Seite steht die Vererbung im Widerspruch zum Geheimnisprinzip. Keine Klasse darf die Attribute einer anderen Klasse sehen.

Es ist wichtig, das die Konzepte der Vererbung sinnvoll eingesetzt werden. Für die Wartbarkeit von Software sind die Aspekte der Generalisierung und Spezialisierung, die durch die Vererbung entstehen, sicher vorteilhaft. In der Definitionsphase können aber Nachteile bezüglich der Lesbarkeit und Verständlichkeit von Strukturen entstehen.

- Verwendete Notation:

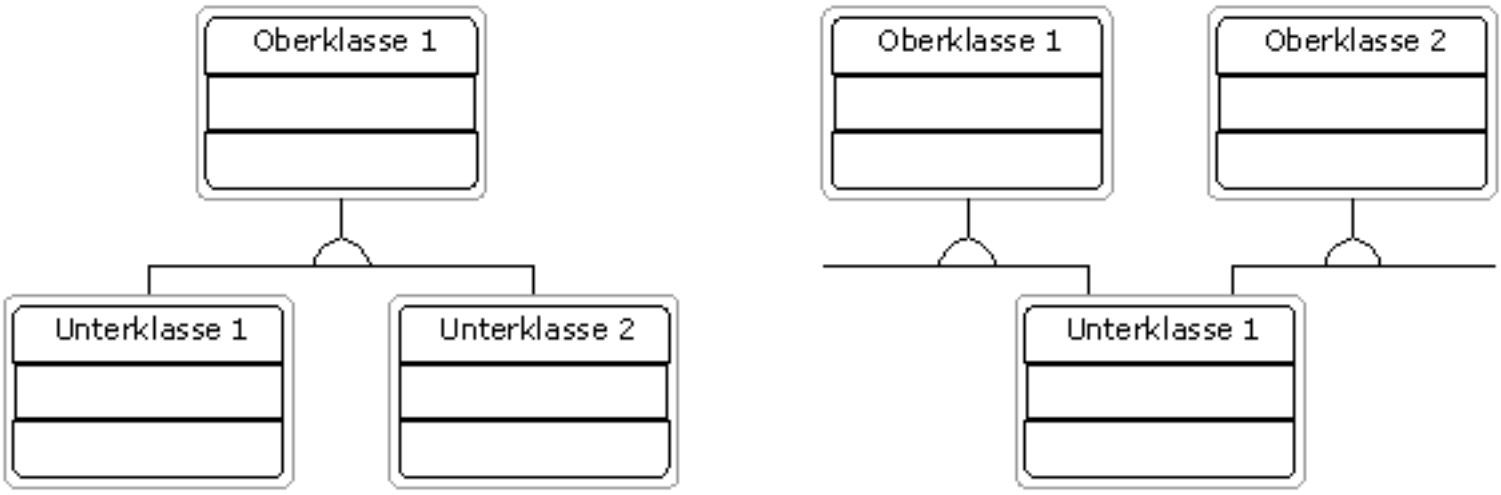


Abbildung 7-5

7.2.6. Polymorphismen

Polymorphismus bedeutet, daß dieselbe Botschaft an Objekte verschiedener Klassen gesendet werden kann, und daß diese unter Umständen darauf verschiedenartig reagieren. Der Sender einer Botschaft muß dann nicht wissen zu welcher Klasse ein Objekt gehört.

Der Polymorphismus erlaubt es, gleichartige Operationen mit dem gleichen Namen zu versehen und auf Objekte verschiedener Klassen anzuwenden. Der Sender muß nur noch wissen, daß ein Empfängerobjekt das gewünschte Verhalten besitzt.

Dieser Ansatz wirkt sich auf Systeme positiv hinsichtlich der Flexibilität und der Änderbarkeit aus. Polymorphismus steht in engem Zusammenhang mit dem Begriff der Vererbung und dem Mechanismus des "späten Binden". Zusätzlich muß es möglich sein redefinierte Operationen zu verwenden.

Durch das Konzept des Polymorphismus wird erreicht, daß jedes Objekt das angemessene Verhalten, hervorgerufen durch eine Botschaft, kennt. Das Objekt kennt seine Klasse implizit. In einer Programmstruktur entfällt die Notwendigkeit von Mehrfachauswahlen zur Gewährleistung eines korrekten Verhaltens.

Der Polymorphismus wird in der objektorientierten Analyse syntaktisch nicht beachtet.

7.2.7. Botschaften

Eine Botschaft ist die Aufforderung eines Senders an einen Empfänger eine Dienstleistung zu erbringen. Die Botschaft wird vom Empfänger interpretiert, und er führt eine entsprechende Operation

durch.

Botschaften bestehen aus dem Namen einer Operation und aus den Argumenten, die für die Operation benötigt werden. Die betreffende Operation kann ihrerseits ein oder mehrere Ergebnisse zurückliefern. Die Menge aller Botschaften, die an Objekte einer Klasse gesendet werden kann, bezeichnet man als Protokoll.

Der Sender einer Botschaft weiß nichts über die Ausführung einer bestimmten Operation. Er interessiert sich nur für die Ergebnisse, die er erhält. Meistens handelt es sich sowohl beim Sender als auch beim Empfänger um ein Objekt. Es besteht aber auch die Möglichkeit, daß Klassen Botschaften austauschen.

Erhält ein Objekt innerhalb einer Vererbungsstruktur eine Botschaft, so prüft es zuerst die eigene Liste der Operationen. Wenn die Operation nicht vorhanden ist, prüft es die Liste der Operationen der direkten Oberklasse.

Im Klassendiagramm wird eine Botschaft durch einen Pfeil zwischen zwei Klassen dargestellt. Er bedeutet, daß eine Kommunikation stattfinden kann, aber nicht unbedingt stattfinden muß.

- Verwendete Notation:

Abbildung 7-6

7.2.8. Assoziationen

Eine Assoziation stellt die Beziehung zwischen Objekten gleichrangiger oder derselben Klasse dar. Sie sind normalerweise bidirektional, müssen aber nicht in beide Richtungen implementiert werden.

Man kann Assoziationen mit den Beziehungsmengen innerhalb eines Entity-Relationship-Modells vergleichen. Es handelt sich dabei jedoch nicht nur um eine statische Struktur, sondern vor allem um die Darstellung des Botschaftsflusses zwischen zwei Objekten. Informationen werden nicht über Schlüssel oder Referenzen erreicht, sie werden von der Assoziation selber geliefert.

Wenn man einer Assoziation eigene Attribute und Operationen zuordnen kann, ist es möglich aus dieser eine eigene Klasse zu modellieren.

In der Regel werden Assoziationen benannt und beschreiben dann meistens nur eine Richtung der

Assoziation. Auf einen Namen kann zum Beispiel verzichtet werden, wenn die Bedeutung der Assoziation offensichtlich ist. Bestehen mehrere Assoziationen zwischen zwei Objekten, so müssen die Assoziationen auf jeden Fall benannt werden.

Assoziationen werden durch eine Linie zwischen zwei Objekten dargestellt.

- Verwendete Notation:

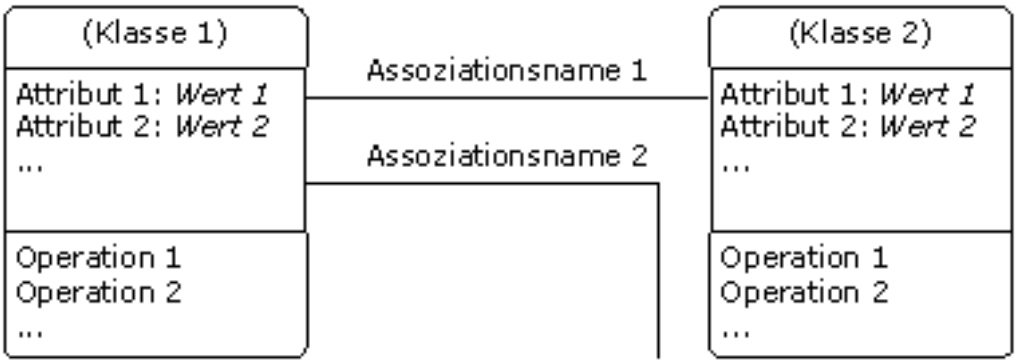


Abbildung 7-7

7.2.9. Kardinalitäten

Die Kardinalität bezieht sich auf eine Assoziation. Sie gibt an, mit wievielen anderen Objekten ein Objekt in Beziehung stehen kann oder muß.

Zur Darstellung kann man die numerische Darstellung benutzen, wobei Zahlen und Buchstaben zur Angabe der minimalen und maximalen Anzahl der Beziehungen verwendet werden. Auf jeder Seite einer Assoziation gibt ein Ziffernpaar, getrennt durch ein Komma an, mit wie vielen anderen Objekten ein Objekt in Beziehung stehen kann. Die erste Ziffer stellt das Minimum dar, die zweite Ziffer das Maximum. Ist das Minimum gleich dem Maximum, so reicht es, nur eine Ziffer zu notieren. Ist die Obergrenze undefiniert, so wird Anstelle einer Zahl ein Platzhalter eingetragen (meistens m oder n).

Die Kardinalität einer Assoziation, die neben einer Klasse eingezeichnet wird, entspricht der möglichen Anzahl der Objekte der anderen Klasse. Entsprechendes gilt für die Kardinalität, die neben der anderen Klasse eingetragen wird.

Es kann zwischen Muß- und Kann-Beziehungen unterschieden werden. Eine minimale Kardinalität von Eins oder mehr weist auf eine Muß-Beziehung hin, eine minimale Kardinalität von Null indiziert eine Kann-Beziehung.

Für spezielle Beziehungen kann man Kardinalitäten auch frei formulieren.

- Verwendete Notation:

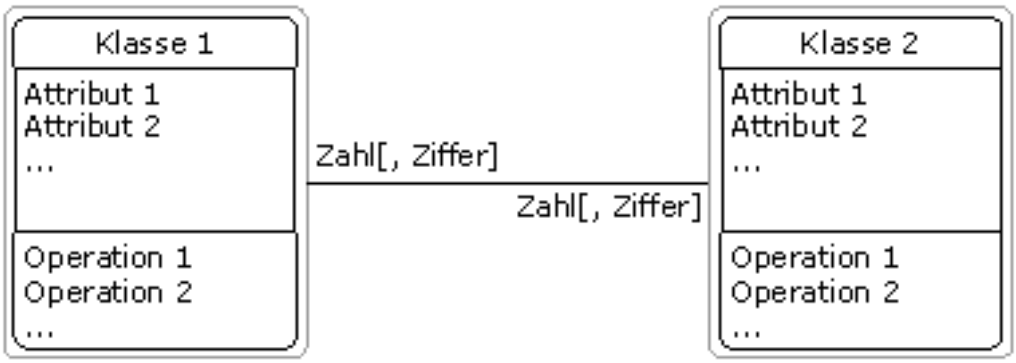


Abbildung 7-8

7.2.10. Rollen

Eine Rolle beschreibt die Funktion eines Objektes innerhalb einer Assoziation. Der Rollenname wird neben der Klasse eingetragen, deren Bedeutung beschrieben wird.

Die Verwendung von Rollen dient in erster Linie der besseren Verständlichkeit von Assoziationen und ist optional. Besteht eine Assoziation zwischen Objekten einer einzelnen Klasse, so ist die Verwendung von Rollen zum Verständnis notwendig.

- Verwendete Notation:

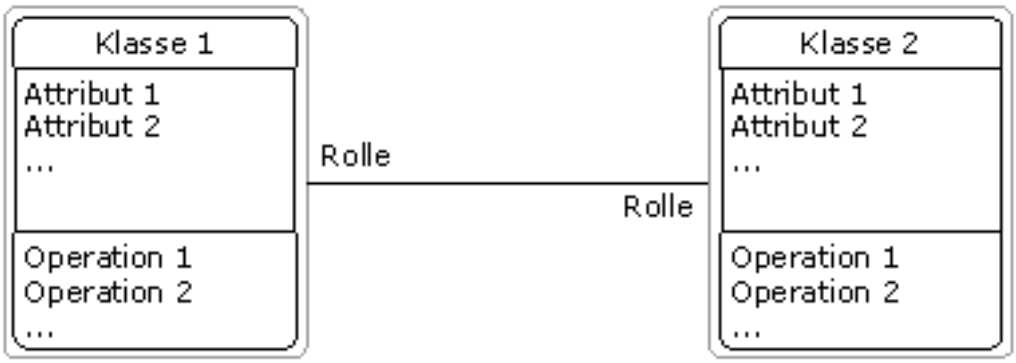


Abbildung 7-9

7.2.11. Restriktionen

Liegt eine gegenseitige Beschränkung zwischen den Attributen zweier Objekte vor, so führt das zu einer Restriktion der Assoziation der beiden Objekte. Darüber hinaus können Assoziationen untereinander restriktiv sein.

Restriktionen werden mit ihrem Namen in einer geschweiften Klammer den entsprechenden Assoziationen zugeordnet. Wenn es der Verständlichkeit dient, werden sie durch einen freien Text

ergänzt.

- Verwendete Notation:

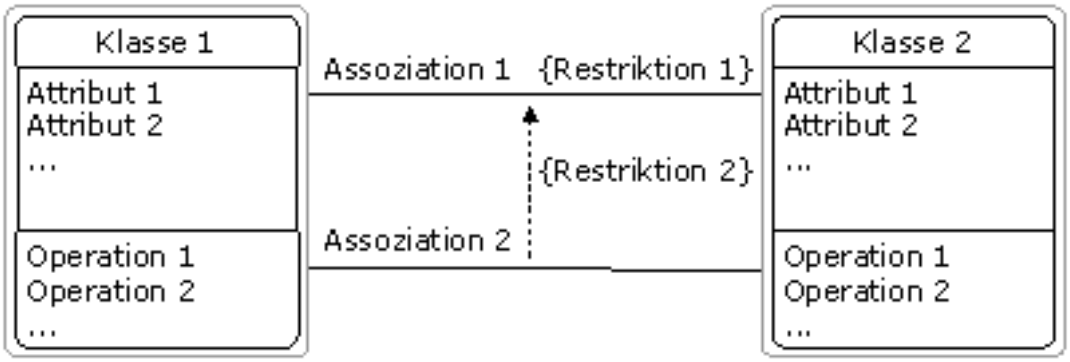


Abbildung 7-10

7.2.12. Aggregationen

Eine Aggregation ist ein Sonderfall einer Assoziation. Sie läßt sich durch eine "Teil-Ganzes-Struktur" darstellen und liegt dann vor, wenn zwischen den Objekten der beteiligten Klassen eine bestimmte Rangordnung gilt.

Ein Objekt, das "das Ganze" darstellt, bezeichnet man als Aggregat-Objekt. Die Existenz von Teil-Objekten kann von der Existenz von Aggregat-Objekten abhängig sein.

Im Normalfall ist eine Aggregation transitiv ($A \in B, B \in C, A \in C$) und asymmetrisch ($A \in B, B \notin A$). Rekursionen (Beziehungen), Kardinalitäten und Restriktionen zwischen Teilobjekt und Aggregatobjekt sind möglich.

Die Teil-Klasse wird durch eine Linie mit der Gesamtheits-Klasse verbunden. Ein Dreieck innerhalb dieser Linie zeigt die Aggregation an (Spitze zur Gesamtheits-Klasse). Auf eine Beschriftung kann verzichtet werden, da grundsätzlich eine "besteht aus"-Beziehung angenommen werden kann. Die Gesamtheits-Klasse sollte in einem Diagramm möglichst über einer Teil-Klasse angeordnet sein.

- Verwendete Notation:

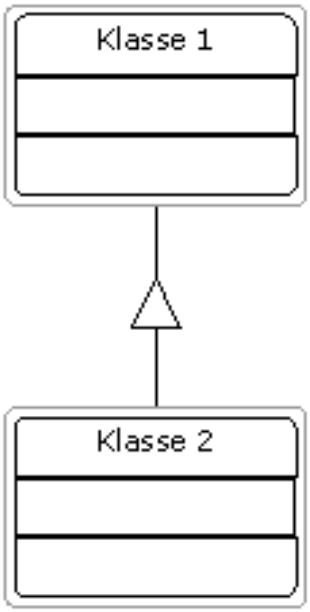


Abbildung 7-11

7.2.13. Subsysteme

Die Einführung von Subsystemen dient bei komplexen Klassenstrukturen einer größeren Übersichtlichkeit und der Findung geeigneter Teilsysteme.

Durch ein Subsystem werden mehrere Klassen zu einer höheren Abstraktionsebene zusammengefaßt. Klassen können zu einem, zu mehreren oder zu keinem Subsystem gehören.

Ein Subsystem sollte folgende Anforderungen erfüllen:

- Der Leser wird besser durch das Gesamtmodell geführt.
- Zusammengehörende Themenbereiche werden zusammengefaßt und können besser verstanden werden.
- Es sind logisch zusammengehörige Klassen enthalten.
- Es kann mit Hilfe von Schnittstellen eigenständig implementiert werden.
- Vererbungsstrukturen werden nicht aufgetrennt.
- Aggregationen sind vollständig enthalten.

Jedes Subsystem wird durch einen eindeutigen Namen gekennzeichnet. Zusätzlich kann man die Subsysteme eines Problembereichs durchnumerieren. Je nach Bedürfnis der Darstellung kann man ein Subsystem in verschiedenen Expansionsformen verkörpern. Die einfachste Form stellt das Subsystem

in einem grauen Rechteck mit seinem Namen dar. Zusätzlich kann noch eine Liste der enthaltenen Klassen aufgeführt werden. Innerhalb des Körpers des Subsystems kann aber auch das komplette Klassenschema des Subsystems eingetragen werden.

- Verwendete Notation:

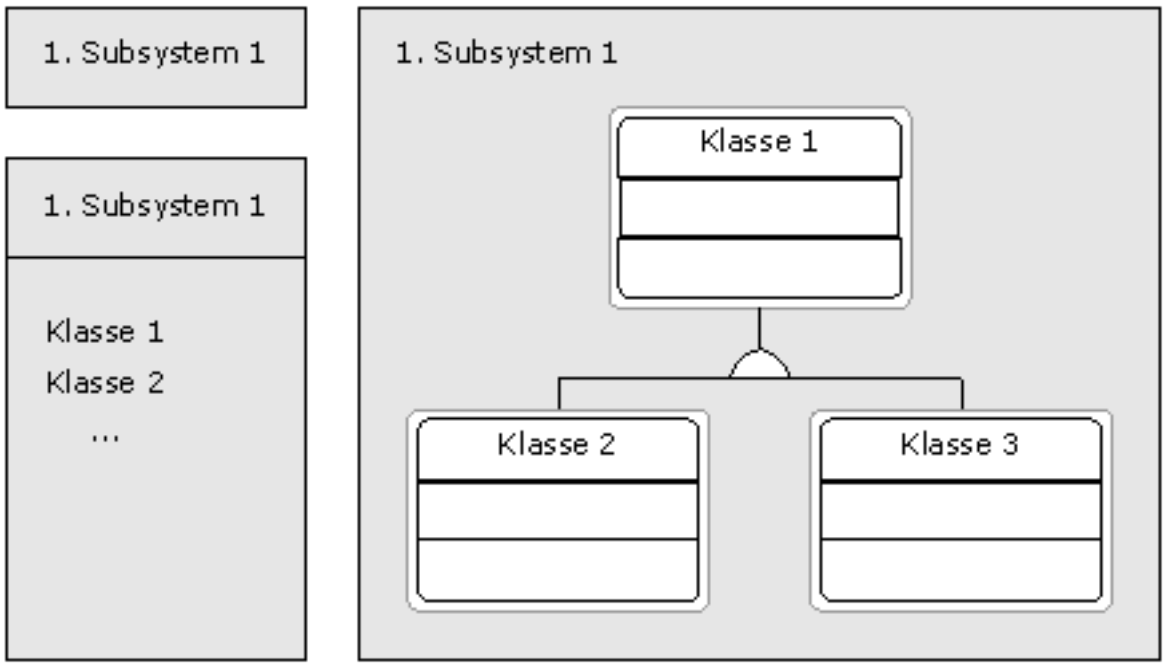


Abbildung 7-12

7.3. Schrittweise Analyse

7.3.1. Einordnung in die bestehende Benutzungsumgebung

Da das Vertriebsinformationssystem nicht als eigenständiges Programm konzipiert ist, ist es vor allen Dingen wichtig, die vorhandenen Schnittstellen zu finden und in die Überlegungen mit einzubeziehen.

Eine erste Darstellung stellt das Vertriebsinformationssystem als Subsystem der "Sage KHK Auftragsbearbeitung" dar, die nur insofern betrachtet wird, wie es für die Überlegungen für die Integration notwendig ist.

Das "Microsoft Office 97" Paket dient auf der einen Seite mit "Microsoft Access 97" als Entwicklungsumgebung und soll auf der anderen Seite durch das Vertriebsinformationssystem enger in die "Sage KHK Auftragsbearbeitung" integriert werden.

Es werden die Schnittstellen zu "Microsoft Outlook 97", "Microsoft Word 97" und "Microsoft Excel 97" in die Modellierung mit einbezogen.

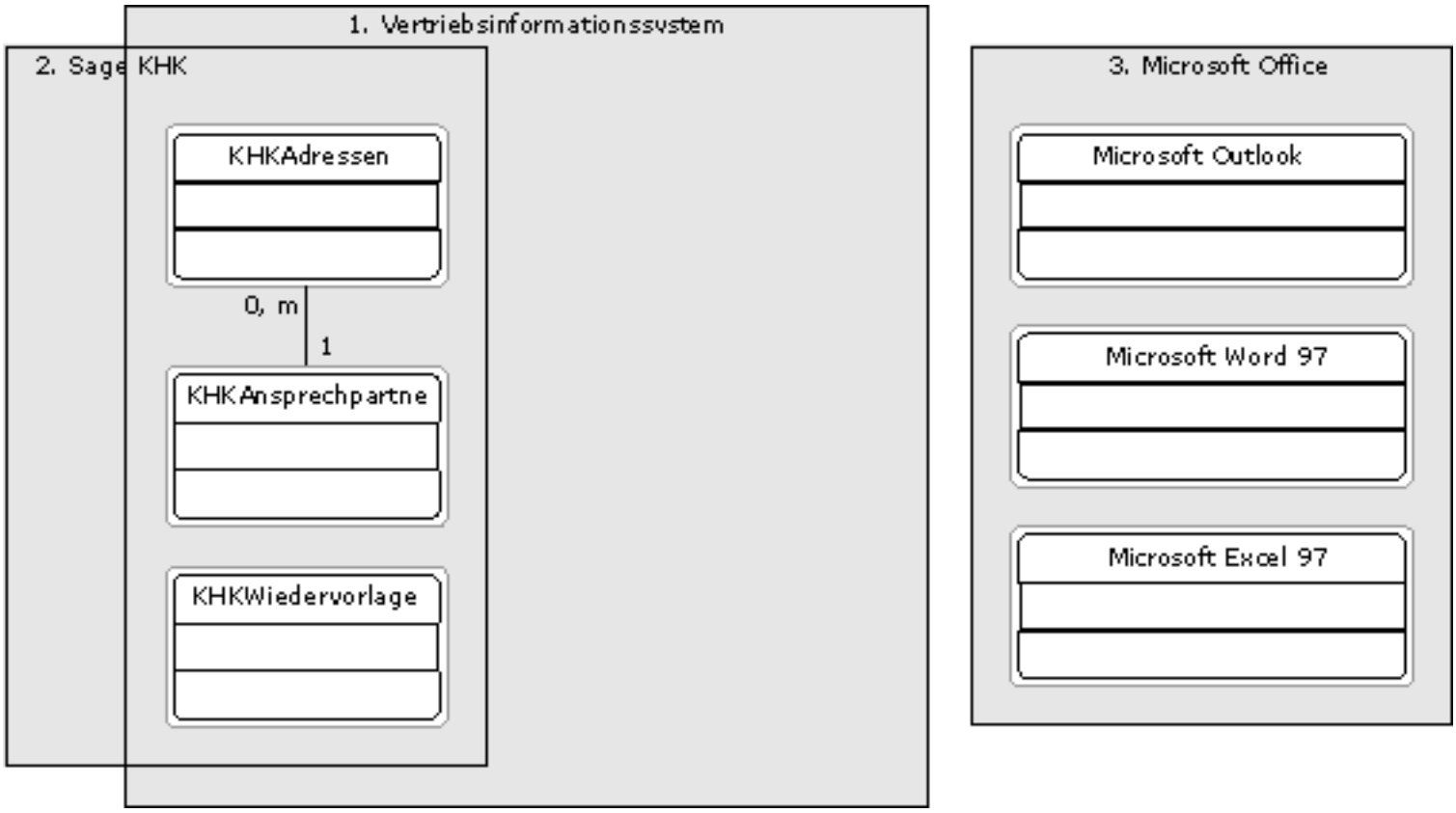


Abbildung 7-13

7.3.2. Finden von Klassen

Klassen zu identifizieren wird von den bestehenden methodischen Ansätzen nicht ausreichend unterstützt. Es gibt viele verschiedene Möglichkeiten der Vorgehensweise, und letztendlich ist ein hohes Maß an Kreativität erforderlich.

Als Ansatzpunkt bietet sich in vielen Fällen das Pflichtenheft an. Substantive, die dort zur Beschreibung der Anforderungen benutzt werden, sind oft Kandidaten für eine neue Klasse. Es besteht allerdings die Gefahr, daß in einem ersten Schritt sehr viele überflüssige Klassen bestimmt werden. Auch kann man nicht davon ausgehen, daß wirklich alle potentiellen Klassen als Substantive im Pflichtenheft auftauchen.

Als Hilfsmittel zum Auffinden von Klassen kann man die "Checkliste zur Identifikation von Klassen" von Heide Balzert nutzen:

- Lassen sich konkrete Objekte identifizieren?

Es gibt keine konkreten Objekte (reale Objekte).

- "Top-down" Vorgehensweise:

Es liegen verbale Anforderungen vor.

Es gibt potentielle Klassen.

Die Klassen besitzen Attribute.

- Zu welchen Kategorien gehören die Klassen?

Informationen über Aktionen.

Allgemeine und fachbezogene Informationen.

- Liegt ein aussagefähiger Klassenname vor?

Substantive mit konkretem Hinweis auf die Bedeutung.

- Welches Abstraktionsniveau liegt vor?

Klassen möglichst geringer Komplexität.

- Liegen 1:1-Beziehungen zwischen Klassen vor?

Es liegt keine 1:1-Beziehung vor.

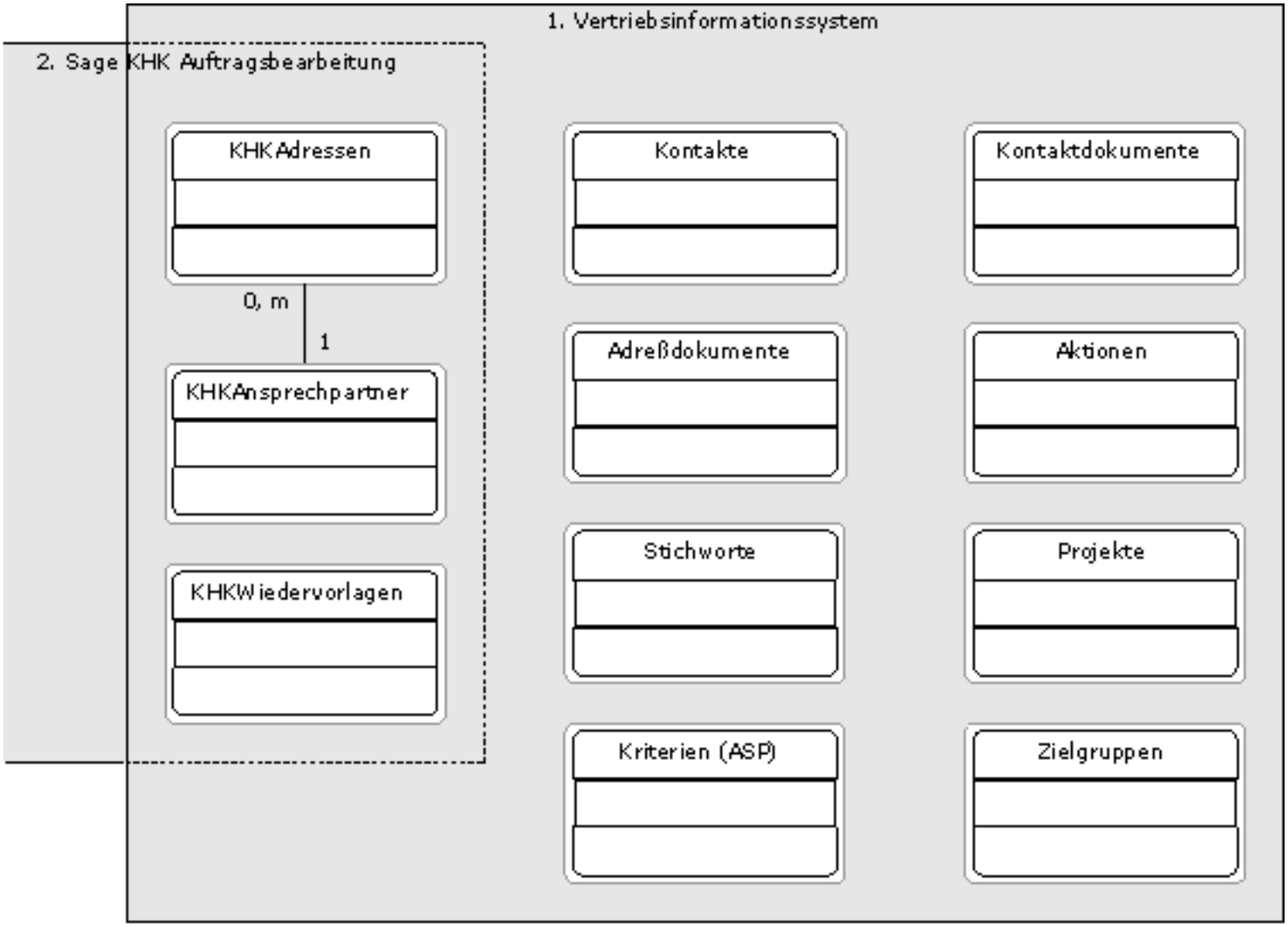


Abbildung 7-14

7.3.3. Finden von Assoziationen, Aggregationen und Kardinalitäten

Ebenfalls mit Hilfe des Pflichtenheftes lassen sich permanente Beziehungen zwischen den Klassen ermitteln.

F25: Zu einem Kunden können beliebig viele Kontakte existieren. Ein Kontakt gehört genau zu einem Kunden (Muß-Beziehung). *F160:* Zu einem Kontakt können beliebig viele Dokumentenverweise gehören. Ein Dokumentenverweis gehört genau zu einem Kontakt (Muß-Beziehung).

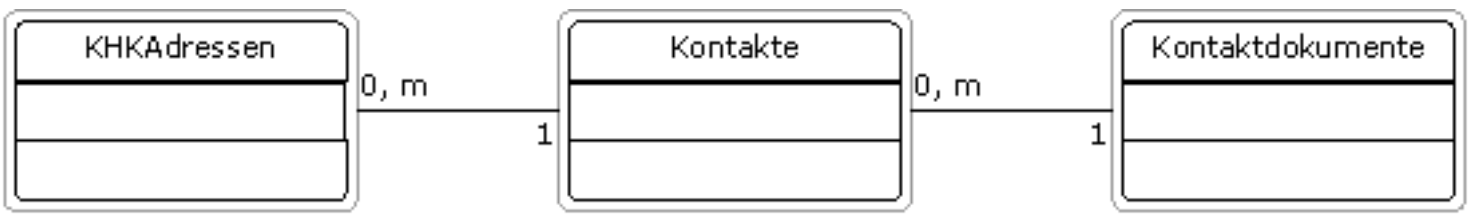


Abbildung 7-15

F30: Zur Selektion von Adressen kann diesen ein oder mehrere Stichworte zugeordnet werden. Ein Stichwort kann auf ein oder mehrere Adressen zutreffen, es kann zunächst aber auch ohne zugehörige Adresse erzeugt werden (Kann-Beziehung).

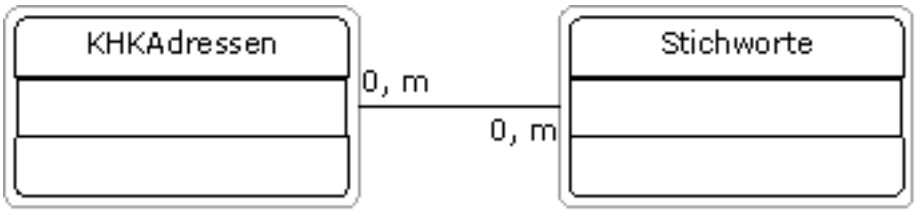


Abbildung 7-16

F50: Einer Adresse können beliebig viele Dokumentenverweise zugeordnet werden. Ein Dokumentenverweis muß genau zu einer Adresse gehören (Muß-Beziehung).

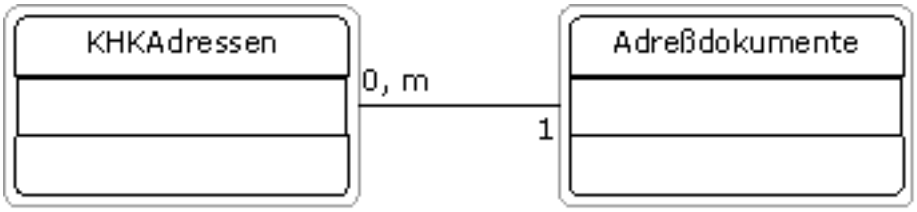


Abbildung 7-17

F120: Einem Ansprechpartner können beliebig viele Kriterien zur Selektion zugeordnet werden. Ein Kriterium kann auf beliebig viele Ansprechpartner zutreffen, es kann aber auch ohne einen zugehörigen Ansprechpartner erzeugt werden (Kann-Beziehung).

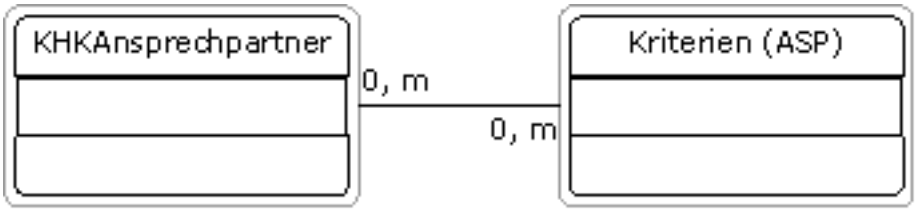


Abbildung 7-18

F250: Zu einer Aktion können beliebig viele Kontakte gehören. Eine Aktion muß keine Kontakte enthalten, Kontakte müssen nicht zu einer Aktion gehören und dürfen höchstens zu einer Aktion gehören (Kann-Beziehung mit Obergrenze).

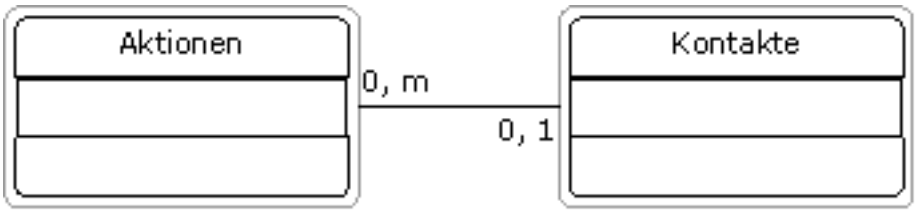


Abbildung 7-19

F290: Zu einem Projekt können beliebig viele Kontakte gehören. Ein Projekt muß keine Kontakte enthalten, Kontakte müssen nicht zu einem Projekt gehören und dürfen höchstens zu einem Projekt gehören (Kann-Beziehung mit Obergrenze).

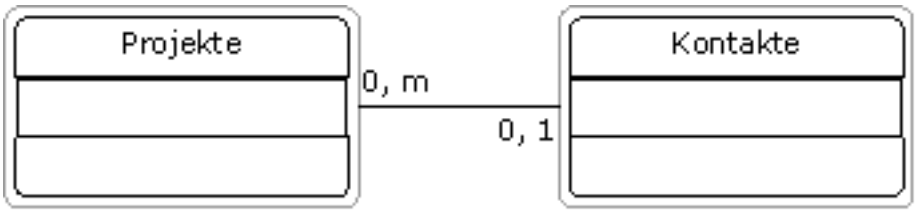


Abbildung 7-20

F350: Zu einer Zielgruppe können beliebig viele SQL-Bedingungen gespeichert werden. Eine Bedingung gehört zu genau einer Zielgruppe (Muß-Beziehung).

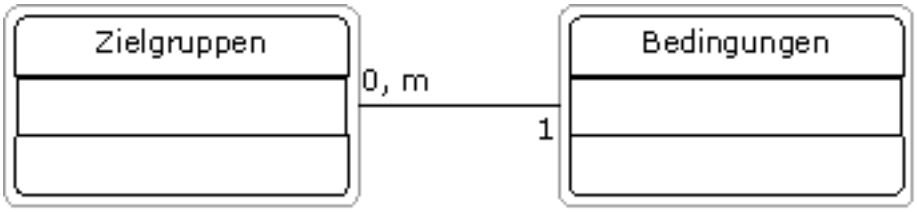


Abbildung 7-21

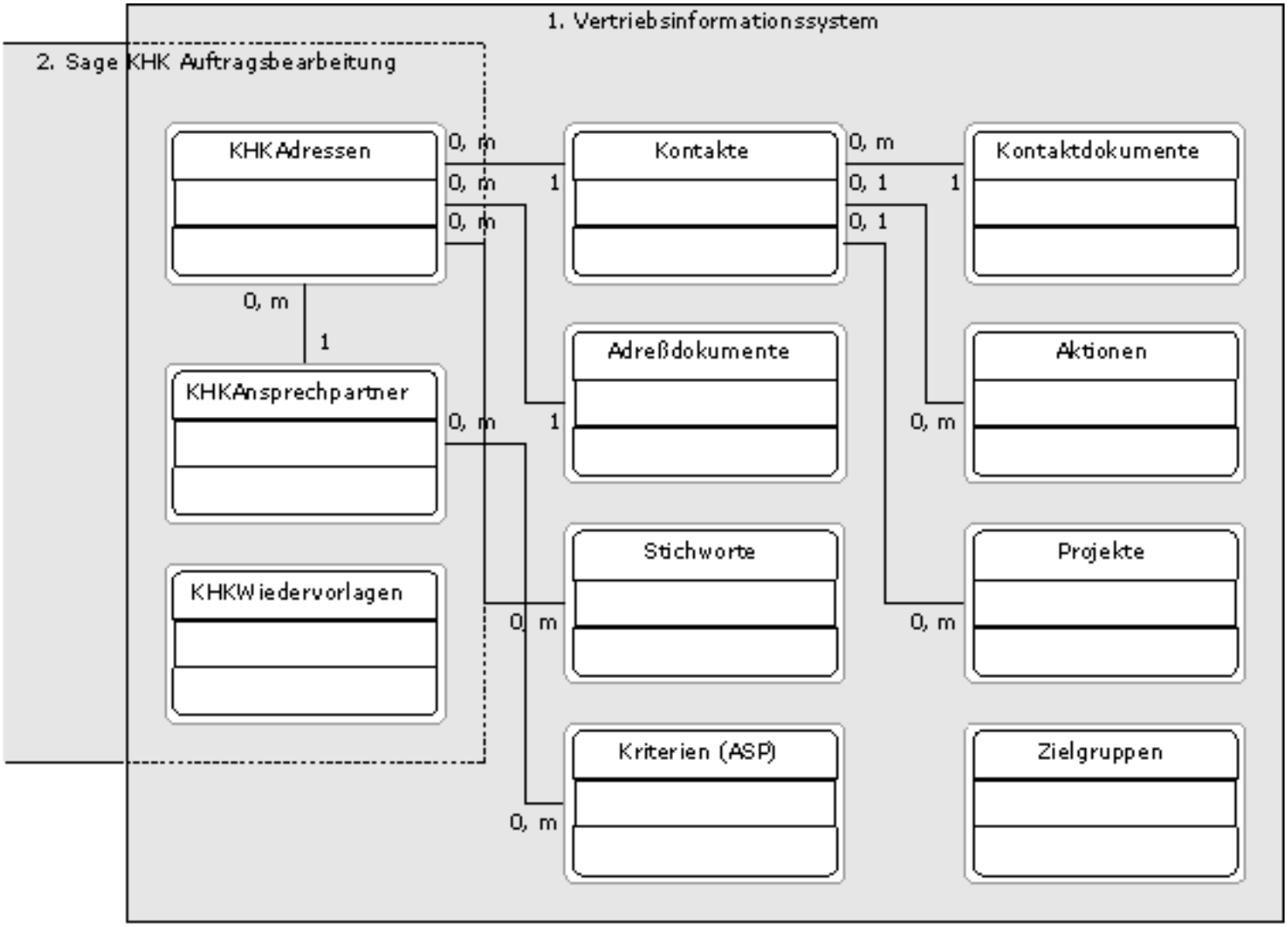


Abbildung 7-22

7.3.4. Finden von Attributen

Auch die Attribute lassen sich im ersten Schritt aus den Angaben aus dem Pflichtenheft ableiten. Der dort definierte Datenbedarf gibt wesentliche Anhaltspunkte für die benötigten Attribute der einzelnen Klassen.

D5: Die Adressen werden um ein Attribut für den zuständigen Vertriebsmitarbeiter im Innendienst, den zuständigen Vertriebsmitarbeiter im Außendienst und den zuständigen Projektleiter erweitert.

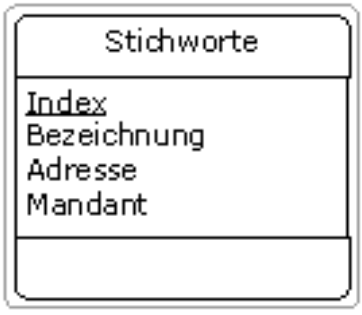
Zuständiger VMI = Typ: VARCHAR(50)

Zuständiger VMA = Typ: VARCHAR(50)

Zuständiger PL = Typ: VARCHAR(50)

Abbildung 7-23

D20: Stichworte dienen der temporären Kennzeichnung von Adressen. Außer der Bezeichnung enthalten sie einen Index, einen Verweis auf die jeweilige Adresse und eine Mandantennummer.



Index = Typ: INTEGER

Schlüssel: Ja

Bezeichnung = Typ: VARCHAR(50)

Adresse = Typ: INTEGER

Mandant = Typ: SMALLINT

Abbildung 7-24

D30: Dokumente zu einer Adresse enthalten einen Verweis auf eine Datei als String, einen Index, einen Verweis auf eine Adresse und eine Mandantennummer.

Index = Typ: INTEGER

Schlüssel: Ja

Dateiverweis = Typ: VARCHAR(200)

Adresse = Typ: INTEGER

Mandant = Typ: SMALLINT

Abbildung 7-25

D40: Die Ansprechpartner werden um die nötigen Attribute erweitert, um mit den "Kontakten" in "Microsoft Outlook 97" konform zu sein.

Vorname = Typ: VARCHAR(20)

Titel = Typ: VARCHAR(20)

Position = Typ: VARCHAR(20)

Beruf = Typ: VARCHAR(50)

Büro = Typ: VARCHAR(50)

Vorgesetzter = Typ: VARCHAR(30)

Sekretär = Typ: VARCHAR(30)

Anrede = Typ: VARCHAR(10)

Briefanrede = Typ: VARCHAR(40)

HauptASP = Typ: BOOLEAN

Persönliche Daten = Typ: STRUCT

Abbildung 7-26

D50: Selektionskriterien für Ansprechpartner enthalten einen Index, eine Bezeichnung, den Namen des Ansprechpartners, einen Verweis auf eine Adresse, einen Verweis auf einen Ansprechpartner und eine Mandantennummer.

Index = Typ: INTEGER

Schlüssel: Ja

Bezeichnung = Typ: VARCHAR(50)

Adresse = Typ: INTEGER

Mandant = Typ: SMALLINT

ASP-ID = Typ: INTEGER

ASP-Name = Typ: VARCHAR(50)

Abbildung 7-27

D60: Ein Kontakt enthält einen Index, ein Datum, eine Uhrzeit, eine Bezeichnung, einen Text, ein Memo, eine Kontaktart, einen Kommentar des Bearbeiters, einen Kommentar des Kunden, ein Kontaktergebnis, den Namen des Ansprechpartners, eine Aktion, ein Projekt, einen Benutzer, einen Verweis auf eine Adresse und eine Mandantennummer.

Index = Typ: INTEGER

Schlüssel: Ja

Datum = Typ: DATE

Uhrzeit = Typ: TIME

Bezeichnung = Typ: VARCHAR(50)

Text = Typ: LONG VARCHAR

Memo = Typ: LONG VARCHAR

Kontaktart = Typ: VARCHAR(50)

Kommentar B = Typ: LONG VARCHAR

Kommentar K = Typ: LONG VARCHAR

Ergebnis = Typ: VARCHAR(50)

ASP-Name = Typ: VARCHAR(50)

Aktion = Typ: INTEGER

Projekt = Typ: VARCHAR(50)

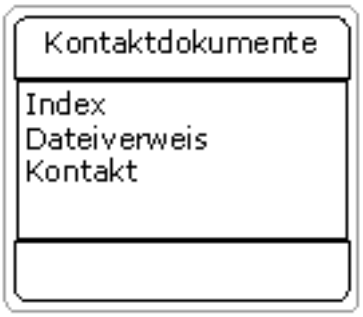
Benutzer = Typ: VARCHAR(4)

Adresse = Typ: INTEGER

Mandant = Typ: SMALLINT

Abbildung 7-28

D90: Kontaktdokumente enthalten einen Index, einen Verweis auf eine Datei als String und einen Verweis auf eine Adresse.



Index = Typ: INTEGER

Schlüssel: Ja

Dateiverweis = Typ: VARCHAR(200)

Kontakt = Typ: INTEGER

Abbildung 7-29

D100: Aktionen enthalten einen Index, eine Bezeichnung, ein Datum, einen Benutzer und eine Mandantennummer.

Index = Typ: INTEGER

Schlüssel: Ja

Bezeichnung = Typ: VARCHAR(100)

Datum = Typ: VARCHAR(10)

Benutzer = Typ: VARCHAR(4)

Mandant = Typ: SMALLINT

Abbildung 7-30

D110: Projekte enthalten einen Index, eine Bezeichnung, einen Anfangstermin, einen Endtermin, einen Status, einen Benutzer und eine Mandantennummer.

Index = Typ: INTEGER

Schlüssel: Ja

Bezeichnung = Typ: VARCHAR(50)

Beginn = Typ: VARCHAR(10)

Ende = Typ: VARCHAR(10)

Status = Typ: VARCHAR(30)

Benutzer = Typ: VARCHAR(4)

Mandant = Typ: SMALLINT

Abbildung 7-31

D120: Zielgruppen enthalten einen Index, eine Bezeichnung, eine SQL-Anweisung, eine Kategorie, eine Gültigkeit, ein Memo, einen Benutzer und eine Mandantennummer.

Index = Typ: INTEGER

Schlüssel: Ja

Bezeichnung = Typ: VARCHAR(50)

SQL-Anweisung = Typ: VARCHAR(200)

Kategorie = Typ: VARCHAR(50)

Gültigkeit = Typ: VARCHAR(10)

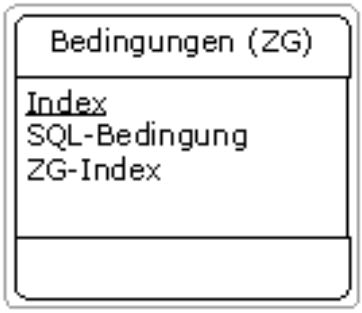
Memo = Typ: LONG VARCHAR

Benutzer = Typ: VARCHAR(4)

Mandant = Typ: SMALLINT

Abbildung 7-32

D130: Zielgruppenbedingungen enthalten einen Index, eine SQL-Bedingung und den Verweis auf eine Zielgruppe.



Index = Typ: INTEGER

Schlüssel: Ja

SQL-Bedingung = Typ: VARCHAR(100)

ZG-Index = Typ: INTEGER

Abbildung 7-33

7.3.5. Finden und spezifizieren von Operationen

Die meisten Operationen werden implizit über die Benutzerschnittstelle initialisiert. Es bietet sich an, das Entwerfen von Bildschirmmasken mit dem Spezifizieren der Operationen zu verzahnen. Als weiterer Ausgangspunkt dient wieder das Pflichtenheft.

Abbildung 7-34 zeigt die Bildschirmmaske zur Bearbeitung von Adressen der "Sage KHK Auftragsbearbeitung". Sie ist um die Elemente des Vertriebsinformations-systems erweitert.

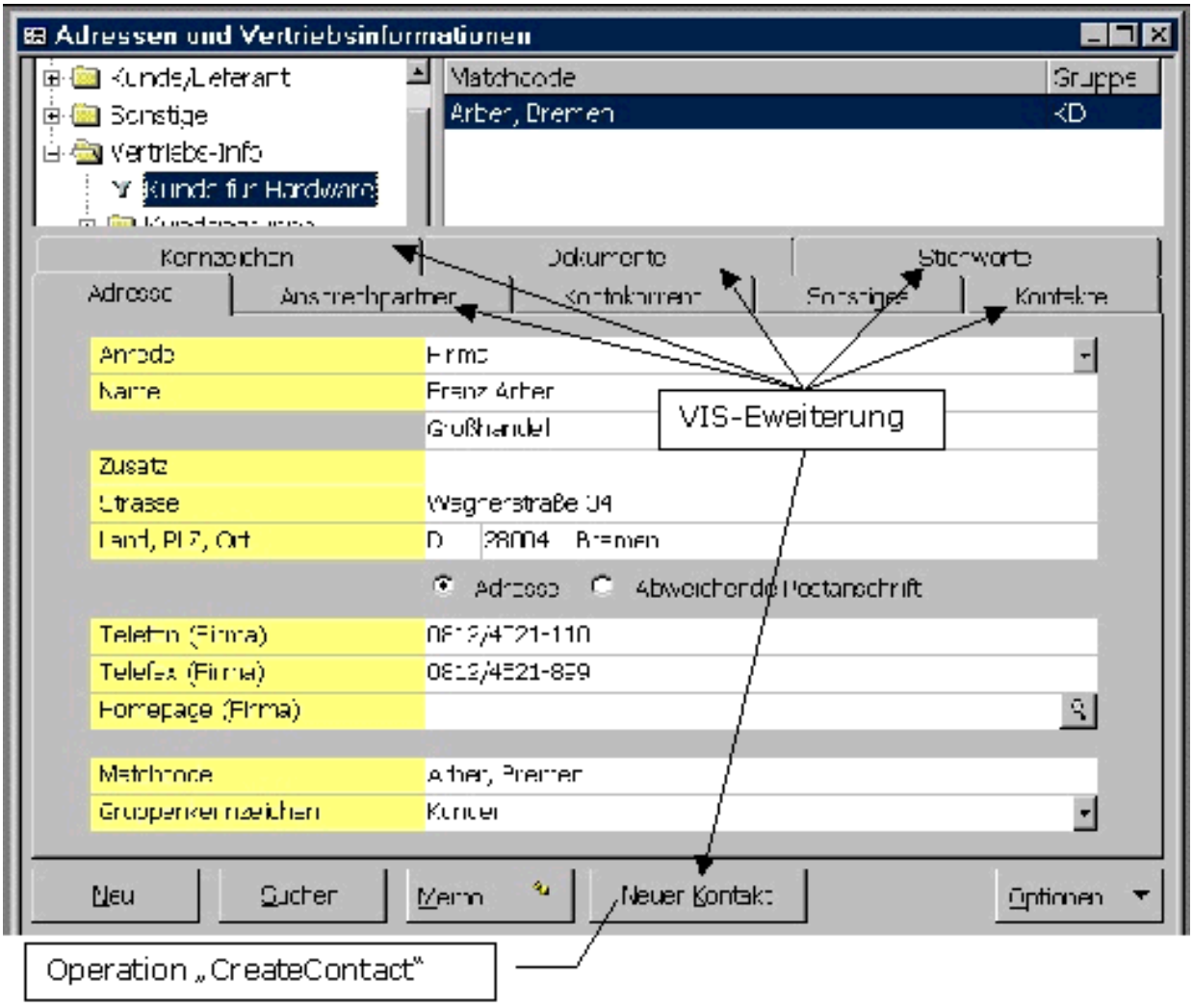


Abbildung 7-34

F160: Ein Kontakt zu einem Kunden wird erzeugt und angezeigt.

Klasse Kontakte

Operation AddContact

```
//Prüfen ob eine Adresse ausgewählt ist

IF NOT IS EMPTY Adresse

THEN

//Erzeugung eines neuen Kontaktes für die ausgewählte Adresse

AddContact(in Parameter1 = KHKAdressen.Adresse)

{
```

```
//Initialisierung des neuen Kontaktes

Kontakte.Index = NEW;

Kontakte.Adresse = Parameter1;

Kontakte.Datum = TODAY;

Kontakte.Uhrzeit = NOW;

Kontakte.Benutzer = CurrentUser;

}

//Anzeige des neuen Kontaktes

SHOW Dialog("Kontaktdetails").Data(in Kontakte.Index);

ELSE

//Fehlermeldung

MESSAGE ("Es ist keine Adresse ausgewählt.");

END IF;
```

Im Register "Kontakte" werden alle Kontakte zu einem Kunden angezeigt. Über einen Button kann man jeden einzelnen anzeigen und bearbeiten.

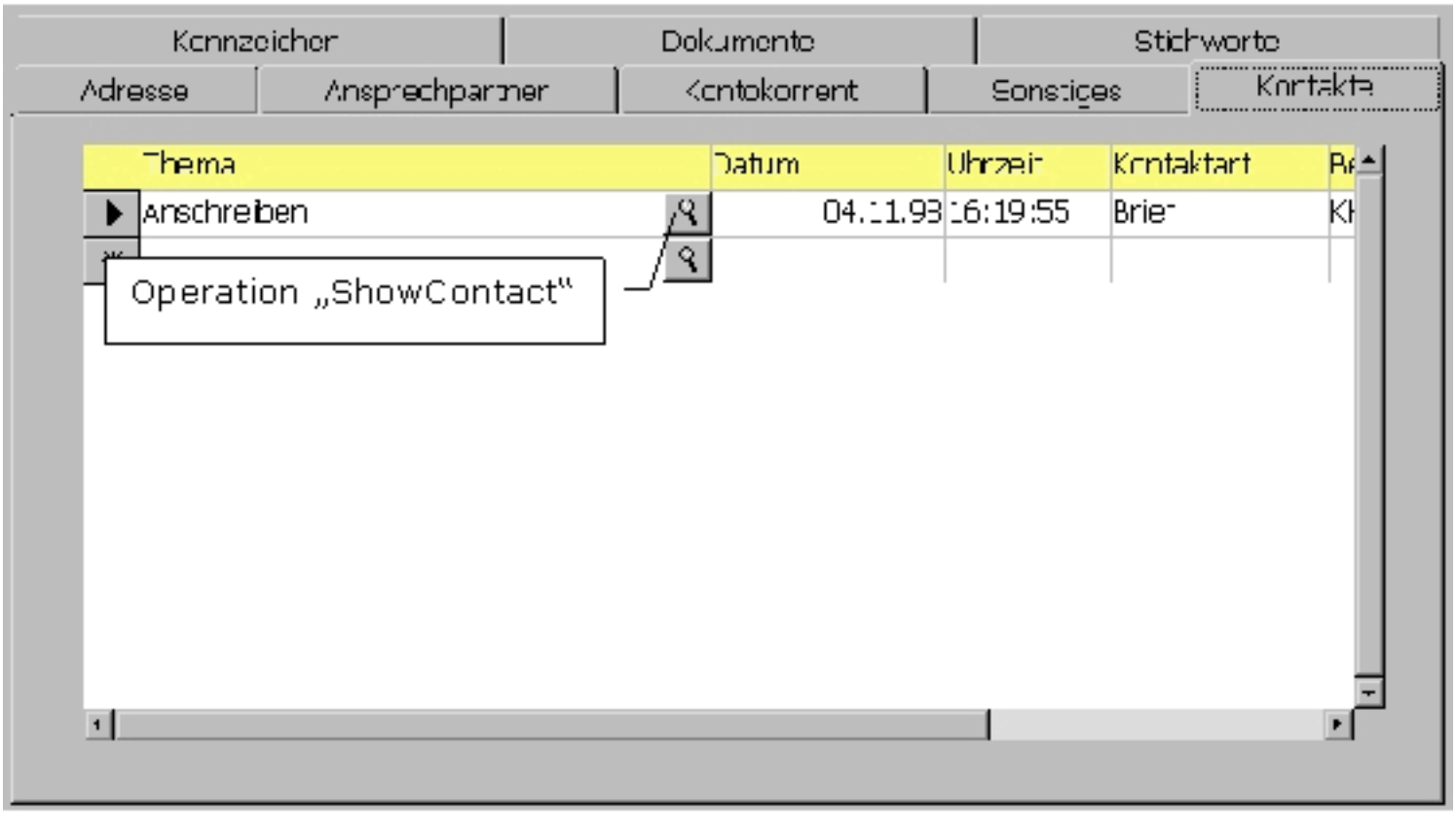


Abbildung 7-35

F210: Anzeigen eines Kontaktes.

Klasse Kontakte

Operation ShowContact

```
//Prüfen ob ein Kontakt ausgewählt ist

IF NOT IS EMPTY Kontakte

THEN

//Anzeige des Kontaktes

ShowContact(in Parameter1 = Kontakte.Index)

{

SHOW DialogKontakte(in Parameter1);

}
```

```
END IF;
```

Kontakte werden im Dialog "Kontaktdetails" (Abbildung 7-36) angezeigt und bearbeitet. Hier werden über einen Menü-Button verschiedene Optionen angeboten (Abbildung 7-37).

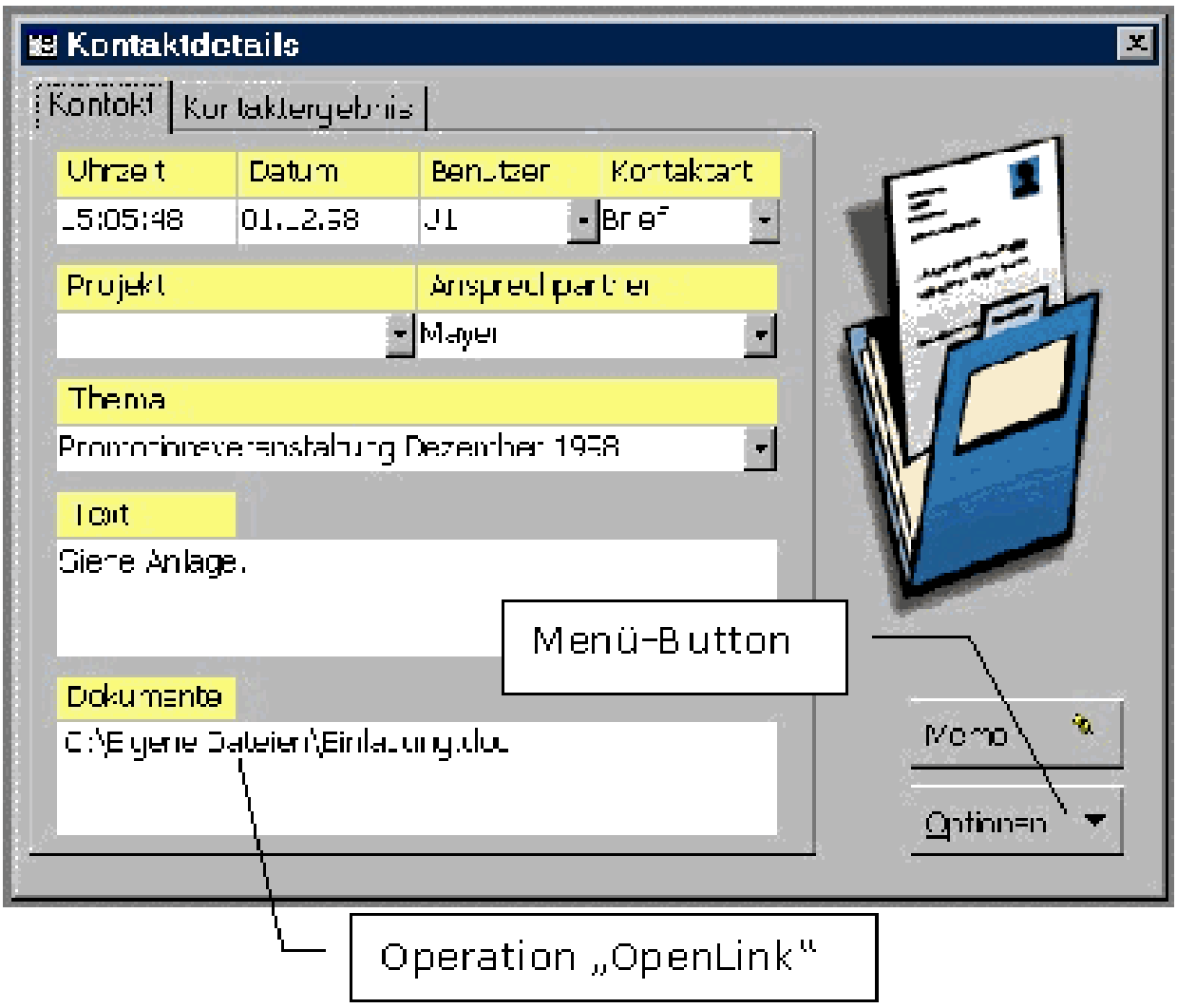


Abbildung 7-36

F170: Durch doppelklicken auf einen Eintrag der Liste "Dokumente" wird die entsprechende Datei mit dem in Windows registrierten Programm geöffnet.

Klasse Kontaktdokumente

Operation OpenLink

```
//Öffnen eines Dokumentverweises

OpenLink(in Kontaktdokumente.Index)
```

```
1

{

//Registriertes Windows-Programm mit dem Verweis öffnen

Windows.Shell(in Kontaktdokumente.Dateiverweis)

}
```

Bei Betätigung des Menü-Buttons erscheint ein Pop-Up Menü (Abbildung 7-37) mit den verschiedenen Optionen, die zur Verfügung stehen.

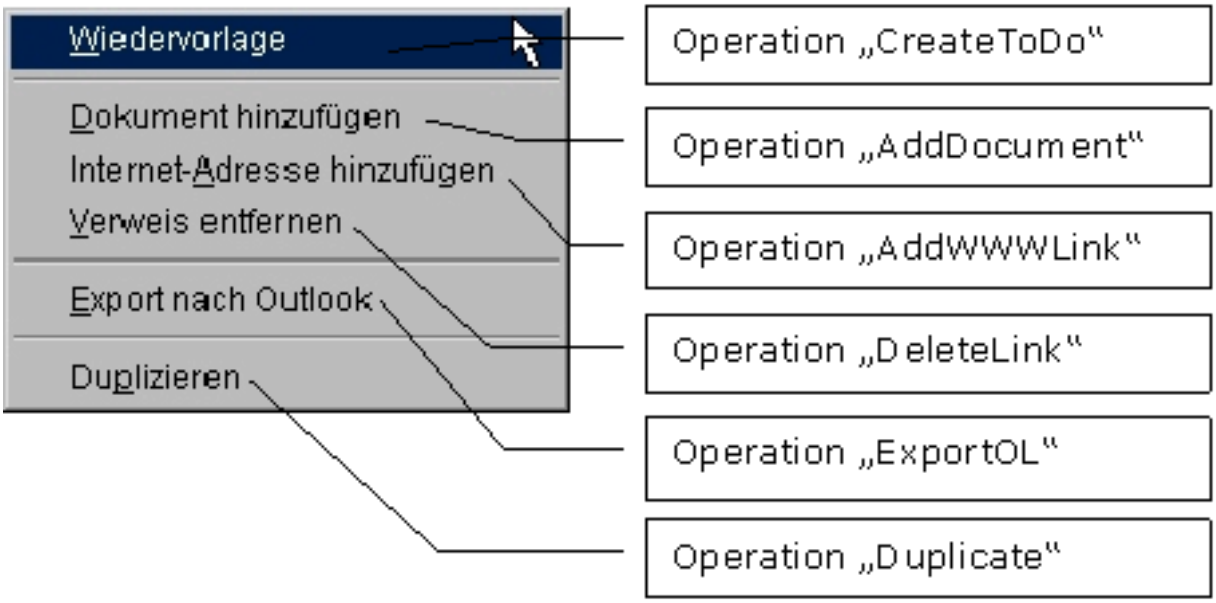


Abbildung 7-37

F180: Übernahme eines Kontaktes in das Wiedervorlagesystem der "Sage KHK Auftragsbearbeitung". Es wird ein Eintrag im Register "Heute zu tun" im Regiezentrum erzeugt.

Klasse Kontakte

Operation CreateToDo

```
//Erzeugung einer neuen Wiedervorlage

CreateToDo(in Parameter1 = Kontakte.Index)

{
```

```
KHKWiedervorlagen.Bezeichnung = Kontakte.Bezeichnung;

KHKWiedervorlagen.Dialog = "Kontaktdetails";

KHKWiedervorlagen.Schlüssel = Parameter1;

}
```

F160: Zu einem Kontakt können beliebig viele Dokumentverweise gespeichert werden.

Klasse Kontaktdokumente

Operation AddDocument

```
//Erzeugung eines Dokumentverweises

AddDocument(in Parameter1 = Kontakte.Index)

{

//Initialisierung des neuen Dokumentverweises

           Kontaktdokumente.Index = NEW;

Kontaktdokumente.Dateiverweis = Windows.FileDialog.File;

Kontaktdokumente.Kontakt = Parameter1;

}

//Liste des Kontaktdialoges aktualisieren

Dialog("Kontaktdetails")Requery;
```

F160: Verweise auf Internet-Seiten werden ebenfalls als Dokumentverweise gespeichert.

Klasse Kontaktdokumente

Operation AddWWWLink

```
//Erzeugung eines Internet-Verweises
```

```
AddWWWLink(in Parameter1 = Kontakte.Index)
{
//Initialisierung des neuen Internet-Verweises
            Kontaktdokumente.Index = NEW;
Kontaktdokumente.Dateiverweis = Windows.Browser.Link;
Kontaktdokumente.Kontakt = Parameter1;
}
//Liste des Kontaktdialoges aktualisieren
Dialog("Kontaktdetails").Requery;
```

F160: Die aufgelisteten Verweise können nach einer Sicherheitsabfrage gelöscht werden.

Klasse Kontaktdokumente

Operation DeleteLink

```
//Löschen eines Verweises
DeleteLink(in Parameter1 = Kontaktdokumente.Index)
{
IF Dialog("Kontaktdetails")Verweisliste.Element.Selected
THEN
//Sicherheitsabfrage
MESSAGE ("Soll der Verweis gelöscht werden?");
IF MESSAGE.Respond = YES
THEN
//Korrespondierenden Verweis löschen
```

```
Kontaktdokumente.Delete(in Parameter1);
END IF;
END IF;
}
//Liste des Kontaktdialoges aktualisieren
Dialog("Kontaktdetails").Requery;
```

F200: Ein Kontakt kann als Aufgabe nach "Microsoft Outlook" exportiert werden.

Klasse Kontakte

Operation ExportOL

```
//Kontakt nach Microsoft Outlook exportieren
ExportOL(in Kontakte.Index)
{
MESSAGE ("Export nach Outlook durchführen?");
IF MESSAGE.Respond = YES
THEN
//Erzeugung des Microsoft Outlook Objektes
Create(out Outlook.Aufgabe)
Outlook.Aufgabe.Betreff = Kontakte.Bezeichnung;
Outlook.Aufgabe.Datum = Kontakte.Datum;
Outlook.Aufgabe.Text = Kontakte.Text;
Outlook.Aufgabe.Fälligkeit = INPUT("Fälligkeit?");
END IF;
```

```
1
}
```

F190: Ein Kontakt kann dupliziert werden.

```
Klasse Kontakte

Operation Duplicate

//Duplizieren eines Kontaktes

Duplicate(in Parameter1 = Kontakte.Index)

{

Kontakte.Index = NEW;

EACH Element OF Kontakte.Index =

            Element OF Kontakte[Parameter1];

            //Duplizierten Kontakt anzeigen

            SHOW DialogKontakte(in Kontakte.Index);

}
```

Es wird ein separates Register für die Ansprechpartner zu einer Adresse hinzugefügt. Über einen Button kann jeder Ansprechpartner in einem Dialog bearbeitet werden.

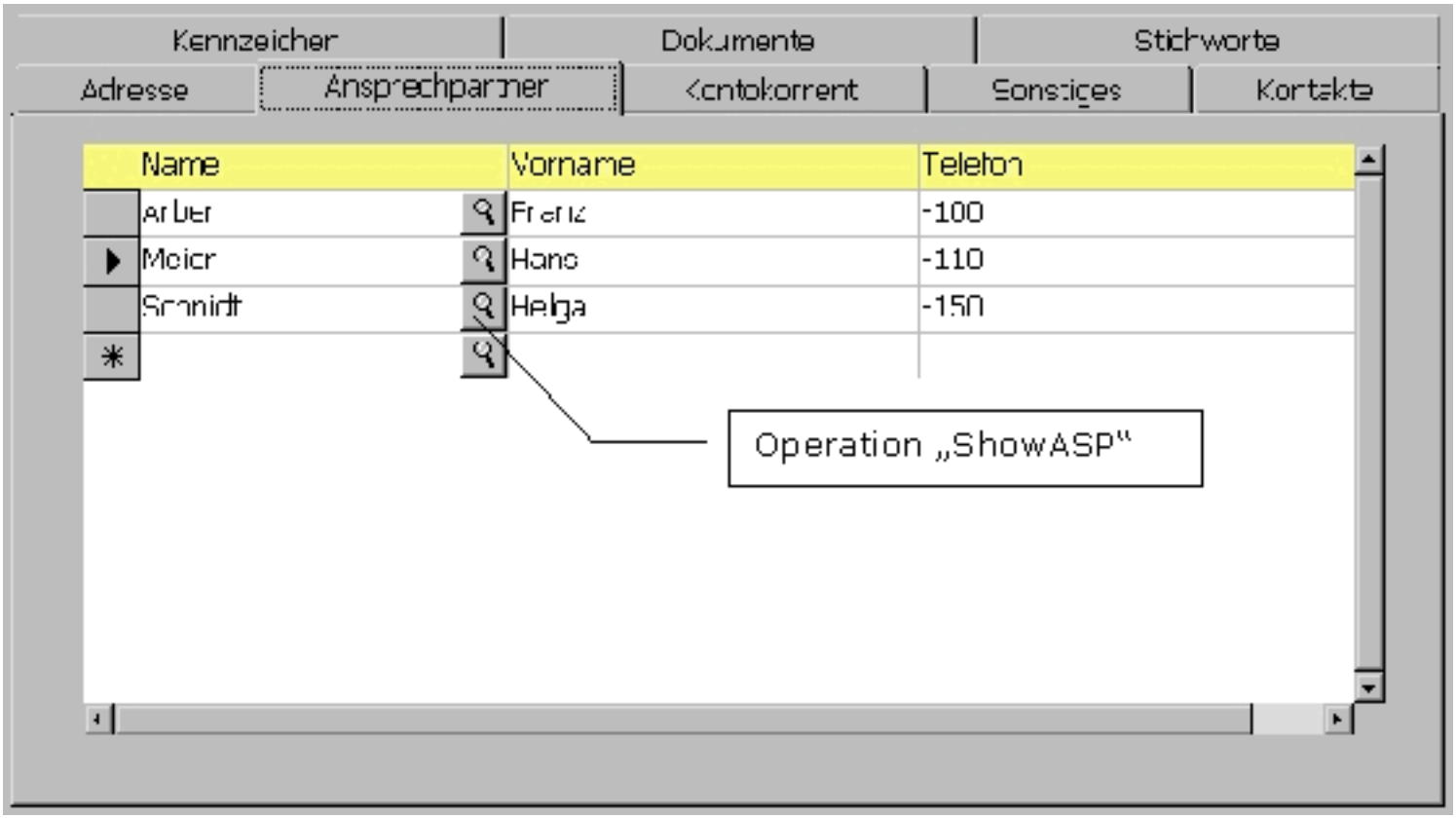


Abbildung 7-38

F110: Anzeigen der Details zu einem Ansprechpartner.

Klasse KHKAnsprechpartner

Operation ShowASP

```
//Prüfen ob ein Ansprechpartner ausgewählt ist

ShowASP(in Parameter1 = KHKAnsprechpartner.Index)

{

           IF NOT IS EMPTY KHKAnsprechpartner

THEN

//Anzeige des Ansprechpartners

SHOW Dialog("Details zum Ansprechpartner").Data(in

                      Parameter1);
```

```
1
            END IF;

}
```

Ansprechpartner werden im Dialog "Details zum Ansprechpartner" (Abbildung 7-39) angezeigt und bearbeitet. Hier werden über einen Menü-Button verschiedene Optionen angeboten (Abbildung 7-40).

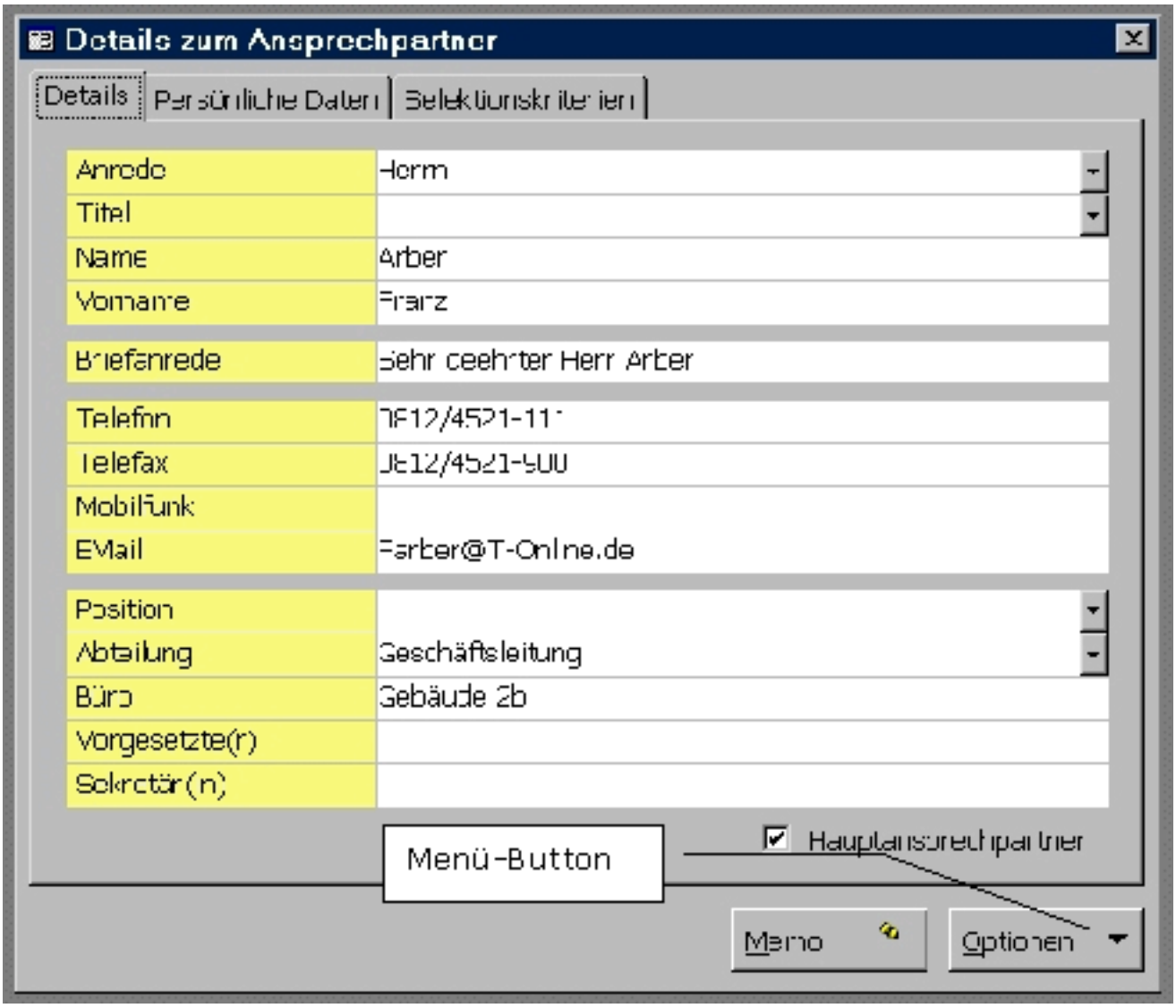


Abbildung 7-39

Bei Betätigung des Menü-Buttons erscheint ein Pop-Up Menü (Abbildung 7-40) mit den verschiedenen Optionen, die zur Verfügung stehen.

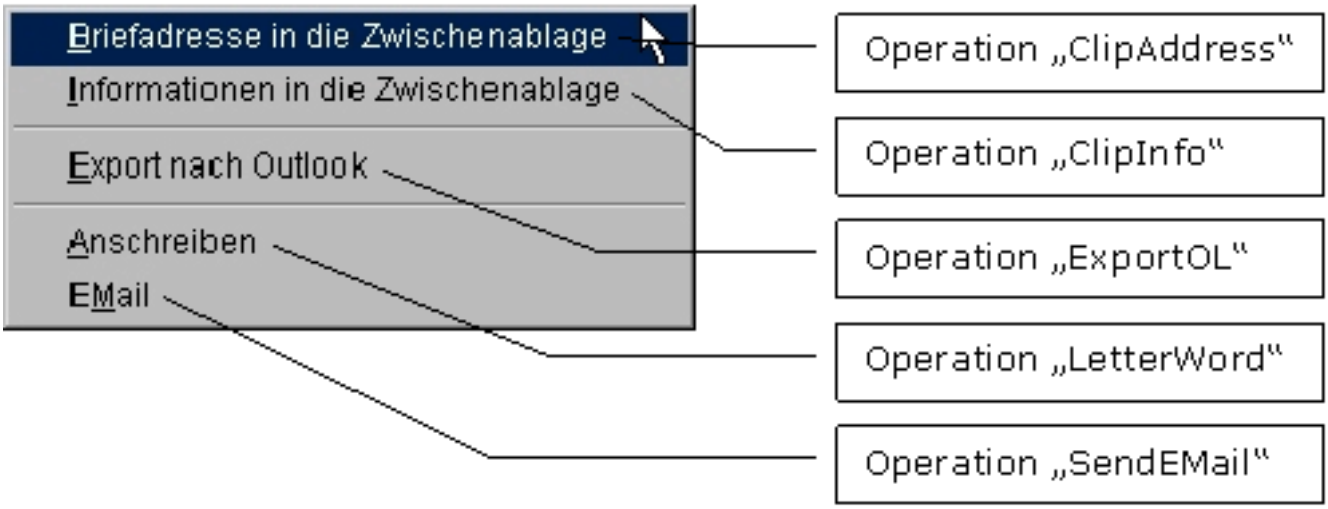


Abbildung 7-40

F140: Die Briefadresse eines Ansprechpartners kann in die Zwischenablage von Windows kopiert werden.

Klasse KHKAnsprechpartner

Operation ClipAddress

```
//Kopieren der Adressdaten eines Ansprechpartners in die

//Zwischenablage

ClipAddress(in Parameter1 = KHKAnsprechpartner.ID)

{

//Zugehörige Adresse ermitteln

            GetAddress(in Parameter1, out Parameter2 =

            KHKAdressen.Adresse);

//Daten formatieren und in die Zwischenablage kopieren

Windows.ClipBoard =

            Format(in Parameter1, in Parameter2, out "Briefadresse");

}
```

F140: Informationen zu einem Ansprechpartner können in die Zwischenablage von Windows kopiert werden.

Klasse KHKAnsprechpartner

Operation ClipInfo

```
//Kopieren von Informationen zu einem Ansprechpartner in die

//Zwischenablage

ClipInfo(in Parameter1 = KHKAnsprechpartner.ID)

{

//Zugehörige Adresse ermitteln

           GetAddress(in Parameter1, out Parameter2 =

           KHKAdressen.Adresse);

//Daten formatieren und in die Zwischenablage kopieren

Windows.ClipBoard =

           Format(in Parameter1, in Parameter2, out "Informationen");

}
```

F140: Ein Ansprechpartner kann in die Adressen von "Microsoft Outlook" übernommen werden.

Klasse KHKAnsprechpartner

Operation ExportOL

```
//Ansprechpartner nach Microsoft Outlook exportieren

ExportOL(in KHKAnsprechpartner.ID)

{

MESSAGE ("Export nach Outlook durchführen?");
```

```
IF MESSAGE.Respond = YES

THEN

//Erzeugung des Microsoft Outlook Objektes

//Ein Outlook-Kontakt ist eine Adresse

                    Create(out Outlook.Kontakt);

EACH CORRESPONDING Element OF Outlook.Kontakt =

                    Element OF KHKAnsprechpartner.ID;

END IF;

}
```

F140: Mit den Daten des ausgewählten Ansprechpartners kann man eine beliebige "Microsoft Word" Vorlage ergänzen. Dort definierte Textmarken werden durch die entsprechenden Daten ergänzt.

Klasse KHKAnsprechpartner

Operation LetterWord

```
//Word-Vorlage öffnen und mit Daten füllen

LetterWord(in Parameter1 = KHKAnsprechpartner.ID)

{

//Zugehörige Adresse ermitteln

          GetAddress(in Parameter1, out Parameter2 =

          KHKAdressen.Adresse);

          //Daten in einem SQL-Objekt zusammenstellen

          SQL-Object(in Parameter1, in Parameter2)

          //Prüfen ob Word geöffnet ist

          IF NOT OPEN("Microsoft Word")
```

```
            THEN
            OPEN("Microsoft Word");
            END IF;
            //Mit der ausgewählten Word-Vorlage wird ein neues
            //Dokument erzeugt
            Word.ActualDocument =
            NEW FROM Windows.FileDialog.File("*.dot");
            //Die Textmarken werden durch entsprechende Daten ersetzt
            EACH CORRESPONDING Element OF Word.ActualDocument.Marks =
            Element OF SQL-Object;
            //Es wird ein Kontakt für den Vorgang erzeugt
            Kontakte.Index = NEW;
            Kontakte.Adresse = Parameter2;
Kontakte.Datum = TODAY;
Kontakte.Uhrzeit = NOW;
            Kontakte.Benutzer = CurrentUser;
            Kontakte.Bezeichnung = "Anschreiben"
            //Das Word-Dokument wird als Dokumentverweis gespeichert
            Kontaktdokumente.Index = NEW;
Kontaktdokumente.Dateiverweis = Windows.FileDialog.File;
            Kontaktdokumente.Kontakt = Kontakte.Index;
}
```

F140: Wenn ein Ansprechpartner eine Email-Adresse hat, kann das registrierte Email-Programm mit dieser Email-Adresse gestartet werden.

Klasse KHKAnsprechpartner

Operation SendEMail

```
//Email-Programm starten und Email-Adresse übergeben

SendEMail(in Parameter1 = KHKAnsprechpartner.ID)

{

//Prüfen ob eine Email-Adresse vorhanden ist

           IF NOT IS EMPTY KHKAnsprechpartner.EMail

THEN

OPEN(MailClient, KHKAnsprechpartner.EMail);

//Zugehörige Adresse ermitteln

                       GetAddress(in Parameter1, out Parameter2 =

                       KHKAdressen.Adresse);

           //Es wird ein Kontakt für den Vorgang erzeugt

                       Kontakte.Index = NEW;

                       Kontakte.Adresse = Parameter2;

Kontakte.Datum = TODAY;

Kontakte.Uhrzeit = NOW;

                       Kontakte.Benutzer = CurrentUser;

                       Kontakte.Bezeichnung = "EMail"

END IF;

}
```

Im Dialog "Details zum Ansprechpartner" kann man einem Ansprechpartner beliebige Kriterien zuordnen, die der Gruppierung und Selektion dienen (Abbildung 7-41).

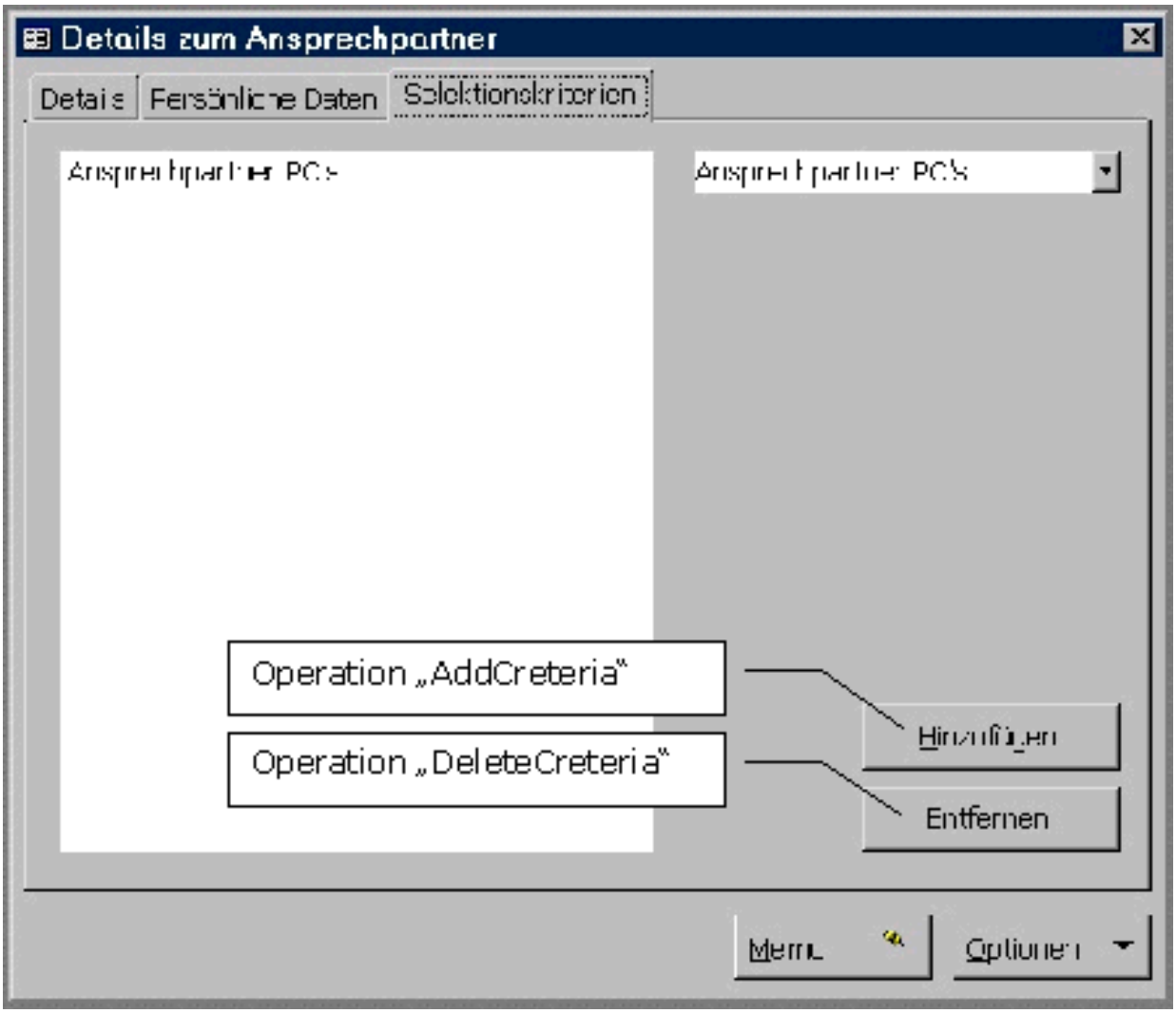


Abbildung 7-41

F120: Einem Ansprechpartner können beliebig viele Selektionskriterien zugeordnet werden.

Klasse KHKAnsprechpartner

Operation AddCreteria

```
//Selektionskriterium zuordnen

AddCreteria(in KHKAnsprechpartner.ID)

{

//Prüfen ob ausgewähltes Kriterium schon vorhanden
```

```
IF NOT IN Dialog("Details zum Ansprechpartner").
           Kriterienliste(in Combo.Text)
THEN
KHKAnsprechpartner.Kriterien.Add(Combo.Text);
//Kriterienliste aktualisieren
                      Dialog("Details zum Ansprechpartner").
                      KriterienListe.Requery;
END IF;
}
```

F120: Die Selektionskriterien können gelöscht werden.

Klasse KHKAnsprechpartner

Operation DeleteCreteria

```
//Selektionskriterium löschen
AddCreteria(in KHKAnsprechpartner.ID)
{
//Prüfen ob ein Kriterium in der Liste ausgewählt ist
IF Dialog("Details zum Ansprechpartner").
           Kriterienliste.Element.Selected
THEN
                      //Sicherheitsabfrage
                      MESSAGE ("Kriterium löschen?");
                      IF MESSAGE.Respond = YES
```

```
KHKAnsprechpartner.Kriterien.Delete(Selected);

//Kriterienliste aktualisieren

                                    Dialog("Details zum Ansprechpartner).

                                    KriterienListe.Requery;

                          END IF;

END IF;

}
```

Im Register "Kennzeichen" (Abbildung 7-42) werden alle Adreßkennzeichen, die im System vorhanden sind aufgelistet. In der Spalte "Wert" erscheinen die Einträge zum ausgewählten Kunden.

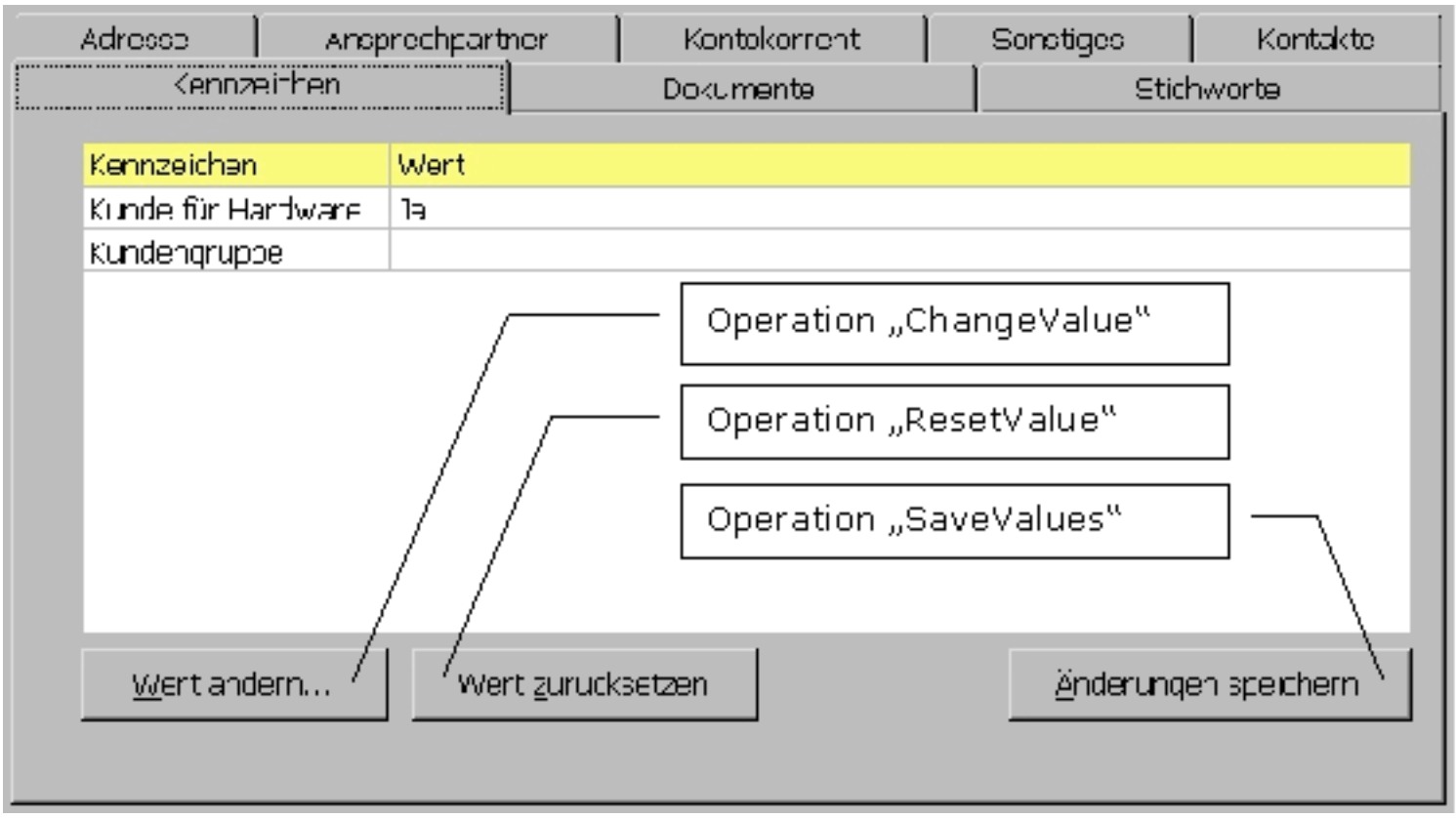


Abbildung 7-42

F10: Die Kennzeichen zu einer Adresse können gesetzt oder geändert werden.

Klasse KHKAdressen

Operation ChangeValue

```
//Kennzeichen ändern oder setzen
ChangeValue
{
//Prüfen, ob ein Kennzeichen ausgewählt ist
           IF Register("Kennzeichen").Liste.Element.Selected
THEN
Register("Kennzeichen").Liste.Element.Selected =
                     INPUT("Wert?");
END IF;
}
```

F10: Die Kennzeichen zu einer Adresse können gelöscht werden.

Klasse KHKAdressen

Operation ResetValue

```
//Kennzeichen löschen
ResetValue
{
//Prüfen, ob ein Kennzeichen ausgewählt ist
           IF Register("Kennzeichen").Liste.Element.Selected
THEN
Register("Kennzeichen").Liste.Element.Selected =
                     EMPTY;
```

```
END IF;

}
```

F10: Die Änderungen werden durch Betätigung eines Buttons in die Datenbank geschrieben.

Klasse KHKAdressen

Operation SaveValues

```
//Alle Werte der Liste in die Datenbank schreiben

SaveValues(in KHKAdressen.Adresse)

{

EACH Kennzeichen IN KHKAdressen =

           Register("Kennzeichen").Liste.Element("Kennzeichen").Wert;

}
```

Im Register "Dokumente" (Abbildung 7-43) werden alle Dokumentverweise der ausgewählten Adresse aufgelistet.

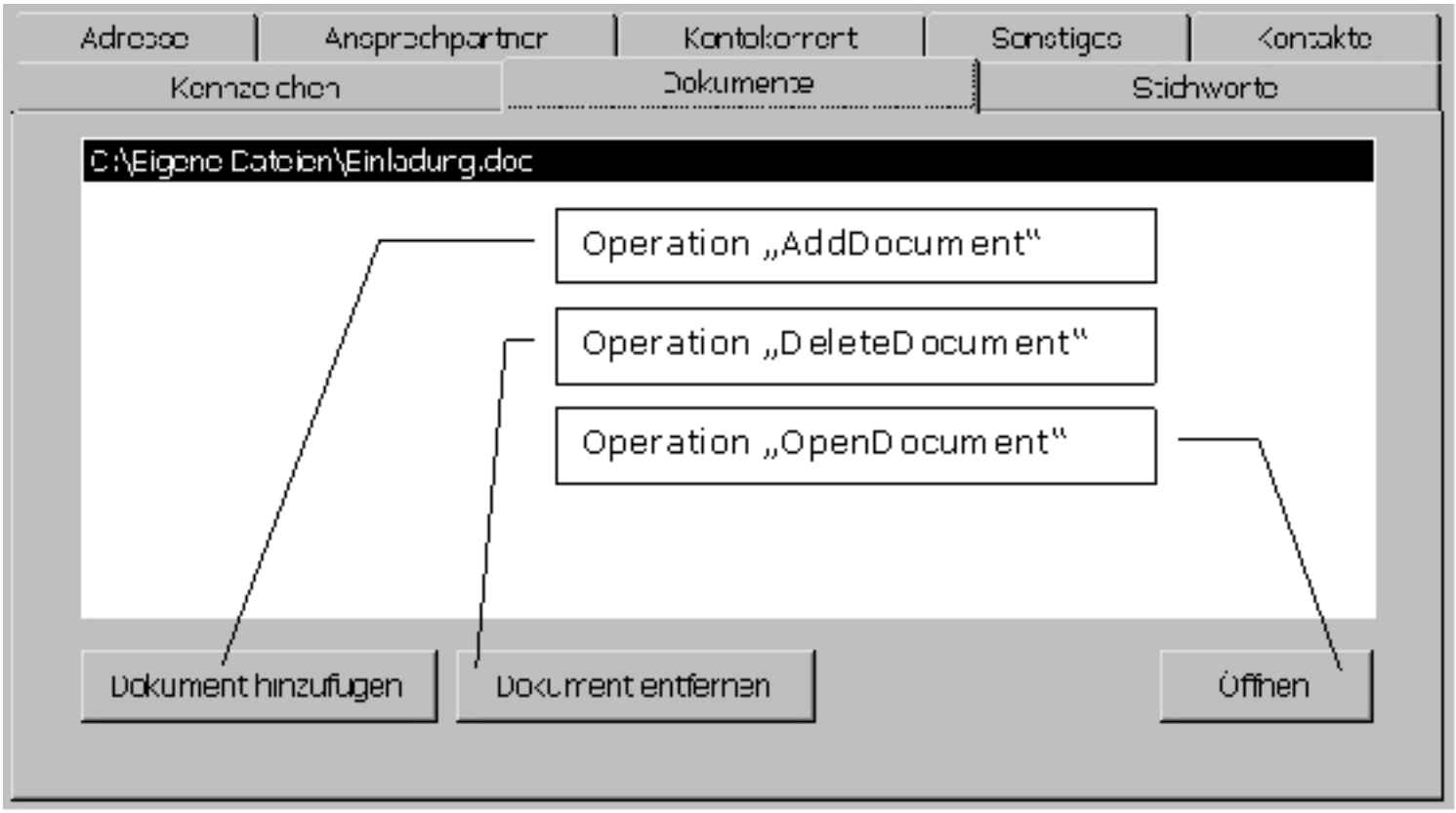


Abbildung 7-43

F50: Einer Adresse können beliebig viele Dokumentverweise hinzugefügt werden.

Klasse Adreßdokumente

Operation AddDocument

```
//Dokumentverweis hinzufügen

AddDocument(in Parameter1 = KHKAdressen.Adresse)

{

//Initialisierung des neuen Dokumentverweises

            Adreßdokumente.Index = NEW;

Adreßdokumente.Dateiverweis = Windows.FileDialog.File;

Adreßdokumente.Adresse = Parameter1;

Adreßdokumente.Mandant = CurrentMandant;

}
```

```
//Liste aktualisieren
Register("Dokumente").Liste.Requery
```

F50: Die Dokumentverweise können nach einer Sicherheitsabfrage entfernt werden.

Klasse Adreßdokumente

Operation DeleteDocument

```
//Dokumentverweis entfernen
DeleteDocument(in Parameter1 = Adreßdokumente.Index)
{
//Prüfen, ob ein Dokumentverweis ausgewählt ist
           IF Register("Dokumente").Liste.Element.Selected
THEN
//Sicherheitsabfrage
MESSAGE ("Soll der Verweis gelöscht werden?");
IF MESSAGE.Respond = YES
THEN
//Korrespondierenden Verweis löschen
Adreßdokumente.Delete(in Parameter1);
END IF;
END IF;
}
//Liste aktualisieren
Register("Dokumente").Liste.Requery
```

F50: Der Dokumentverweis kann mit dem registrierten Windows-Programm geöffnet werden.

Klasse Adreßdokumente

Operation OpenDocument

```
//Öffnen eines Dokumentverweises

OpenDocument(in Adreßdokumente.Index)

{

//Registriertes Windows-Programm mit dem Verweis öffnen

Windows.Shell(in Kontaktdokumente.Dateiverweis)

}
```

Im Register "Stichworte" (Abbildung 7-44) werden alle Stichworte der ausgewählten Adresse aufgelistet. Sie dienen einer temporären Kennzeichnung einer Adresse.

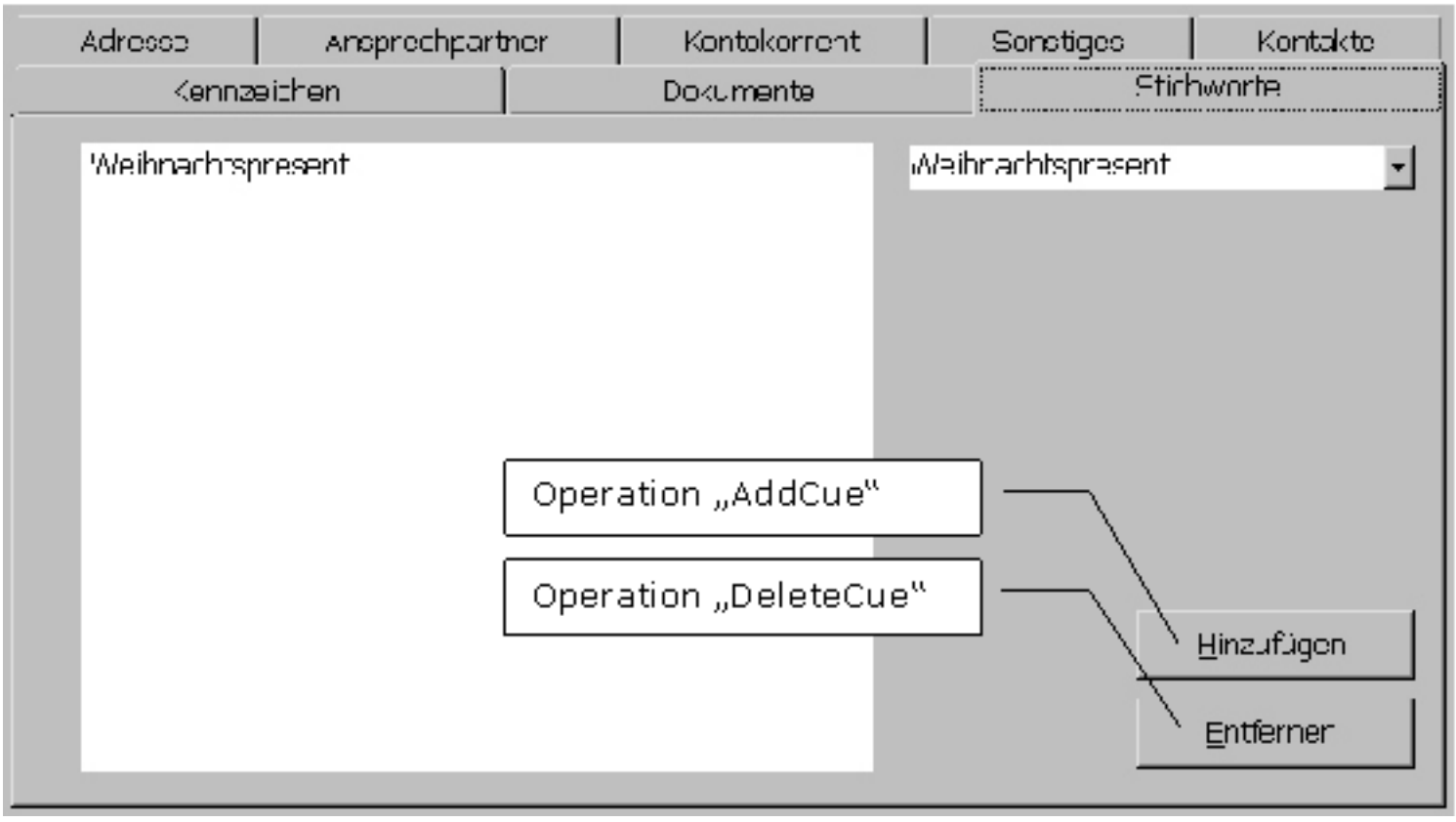


Abbildung 7-44

F30: Hinzufügen eines Stichwortes zu einer Adresse.

Klasse Stichworte

Operation AddCue

```
//Stichwort hinzufügen
AddCue(in Parameter1 = KHKAdressen.Adresse)
{
//Prüfen, ob ausgewähltes Stichwort schon vorhanden ist
IF NOT IN Register("Stichworte").Liste(in Combo.Text)
THEN
                        //Initialisierung des neuen Stichwortes
                        Stichworte.Index = NEW;
Stichworte.Bezeichnung = Combo.Text;
                        Stichworte.Adresse = Parameter1;
                        Stichworte.Mandant = CurrentMandant;
END IF;
}
//Liste aktualisieren
Register("Stichworte").Liste.Requery;
```

F30: Stichwort können nach einer Sicherheitsabfrage wieder entfernt werden.

Klasse Stichworte

Operation DeleteCue

```
//Stichwort entfernen

DeleteCue(in Parameter1 = Stichworte.Index)

{

//Prüfen, ob ein Stichwort ausgewählt ist

           IF Register("Stichworte").Liste.Element.Selected

THEN

//Sicherheitsabfrage

MESSAGE ("Soll das Stichwort gelöscht werden?");

IF MESSAGE.Respond = YES

THEN

//Korrespondierendes Stichwort löschen

Stichworte.Delete(in Parameter1);

END IF;

END IF;

}

//Liste aktualisieren

Register("Stichworte").Liste.Requery
```

Im Dialog "Ansprechpartner" (Abbildung 7-45) können Ansprechpartner nach verschiedenen Kriterien gesucht werden. Wählt man einen Ansprechpartner aus, wird die zugehörige Firma angezeigt, und man kann sich Details anzeigen lassen (Abbildung 7-39).

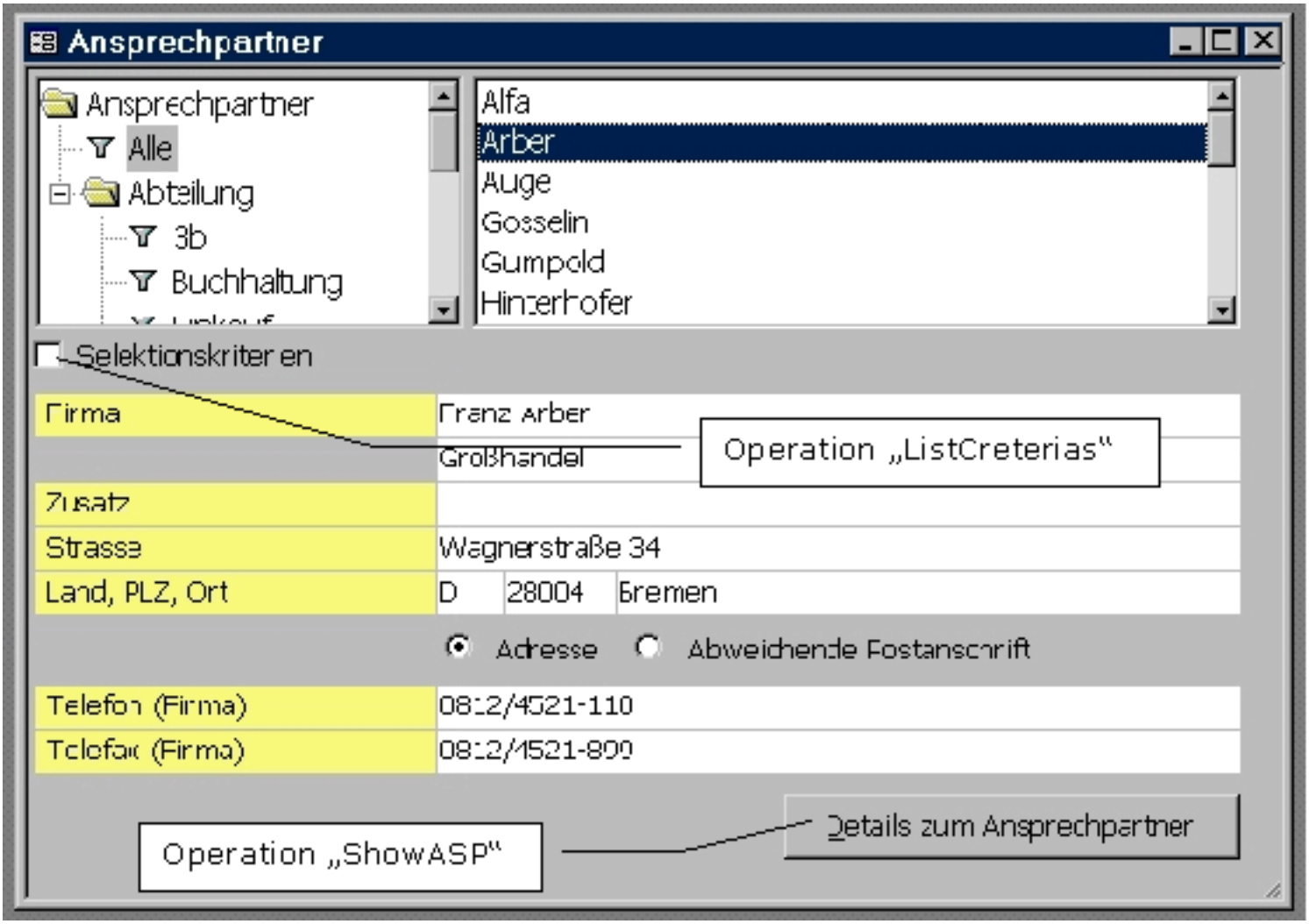


Abbildung 7-45

F100: Die Ansprechpartner können nach den zugewiesenen Selektionskriterien aufgelistet werden.

Klasse KHKAnsprechpartner

Operation ListCreteria

```
//Selektionskriterien auflisten

ListCreteria

{

//Der Daten-Browser wird für die Selektionskriterien

          //initialisiert

Formular("Ansprechpartner").DataBrowser =
```

```
            KHKAnsprechpartner.Selektionskriterien;

}

//Die Anzeige des Formulars wird aktualisiert

Formular("Ansprechpartner").Requery
```

F110: Anzeigen der Details zu einem Ansprechpartner.

Klasse KHKAnsprechpartner

Operation ShowASP

```
//Prüfen ob ein Ansprechpartner ausgewählt ist

ShowASP(in Parameter1 = KHKAnsprechpartner.Index)

{

            IF NOT IS EMPTY KHKAnsprechpartner

THEN

//Anzeige des Ansprechpartners

SHOW Dialog("Details zum Ansprechpartner").Data(in

                        Parameter1);

            END IF;

}

//Es handelt sich um den gleichen Dialog wie in Abbildung 7-39
```

Abbildung 7-46 zeigt die Bildschirmmaske für alle verkaufsbezogenen Vorgänge der "Sage KHK Auftragsbearbeitung". Sie ist um das Register "Kontakte" erweitert.

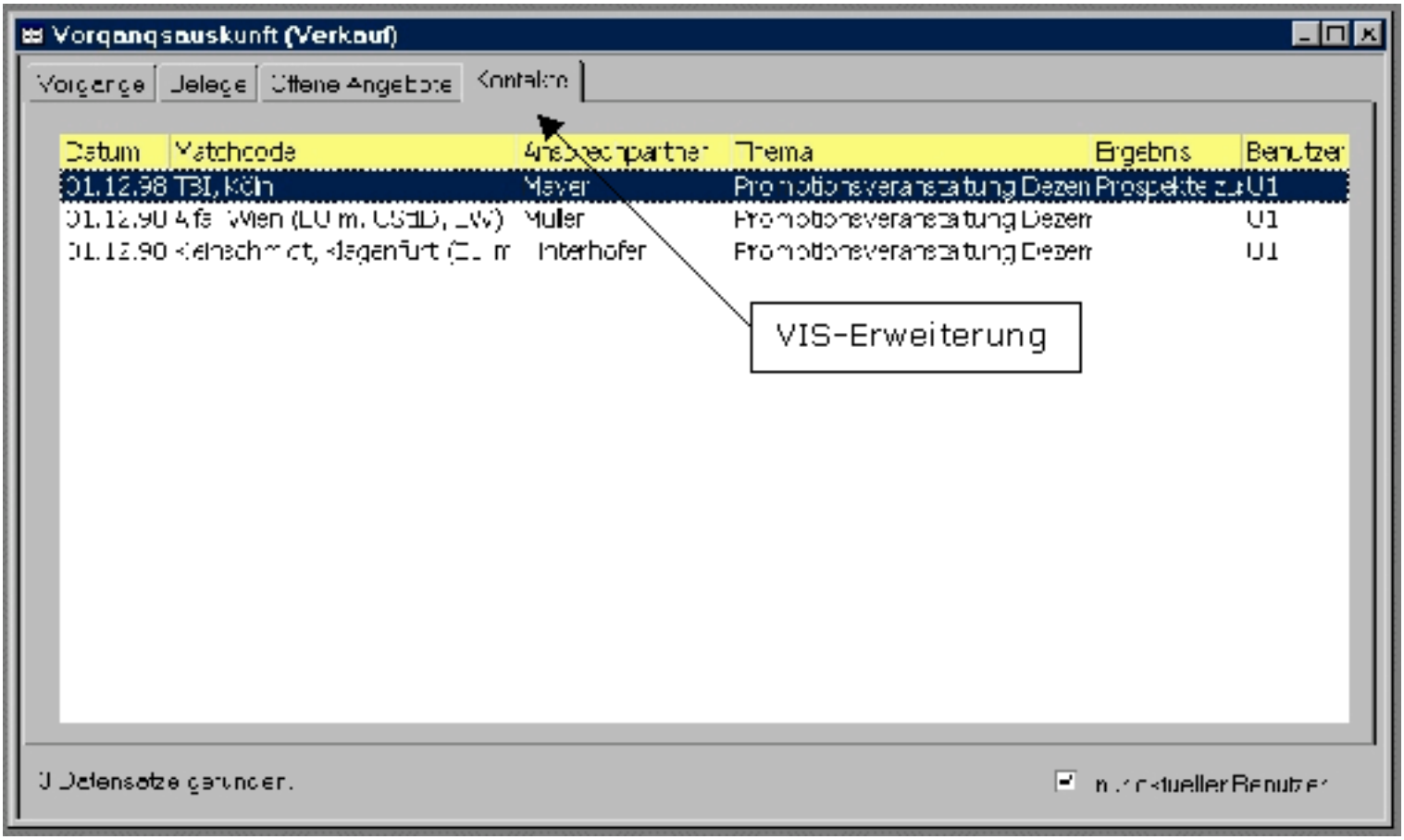


Abbildung 7-46

Wählt man einen Kontakteintrag aus, hat man mit der rechten Maustaste ein kontextsensitives Menü zur Verfügung.

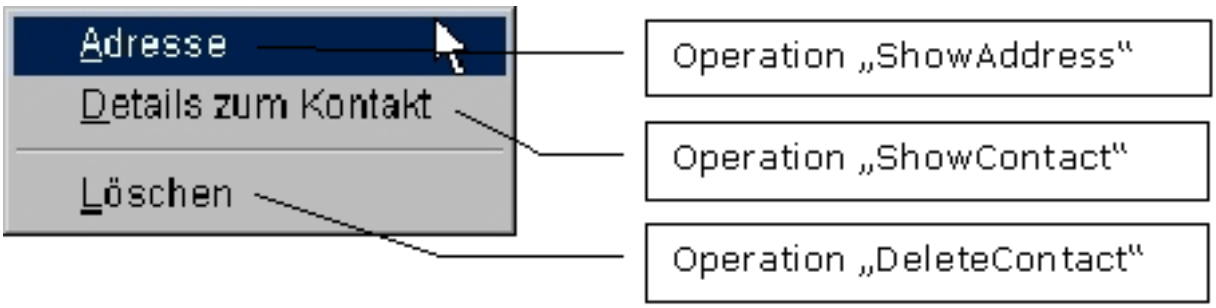


Abbildung 7-47

F230: Anzeigen der zugehörigen Adresse.

Klasse Kontakte

Operation ShowAddress

```
//Anzeige der Adresse

ShowAddress(in Parameter1 = Kontakte.Index)

{

//Zugehörige Adresse ermitteln

            GetAddress(in Parameter1, out Parameter2 =

            KHKAdressen.Adresse);

                        SHOW Dialog("Adressen und
                        Vertriebsinformationen").Data(in Parameter2);

}
```

F230: Anzeigen eines Kontaktes.

Klasse Kontakte

Operation ShowContact

```
//Anzeige des Kontaktes

ShowContact(in Kontakte.Index)

{

SHOW DialogKontakte(in PARAMETER1);

}
```

F230: Löschen eines Kontaktes.

Klasse Kontakte

Operation ShowAddress

```
//Löschen des Kontaktes

DeleteContact(in Parameter1 = Kontakte.Index)
```

```
1

{

//Sicherheitsabfrage

            MESSAGE ("Soll der Kontakt gelöscht werden?");

IF MESSAGE.Respond = YES

THEN

//Ausgewählten Kontakt löschen

                        DELETE(Parameter1)

END IF;

}
```

Im Dialog "Zielgruppen" (Abbildung 7-48) können Zielgruppen angelegt, geändert, gelöscht oder ausgewählt werden. Im Register "Zielgruppe" kann eine Bezeichnung, ein Benutzer, eine Kategorie, eine Gültigkeit und ein Memo hinterlegt werden.

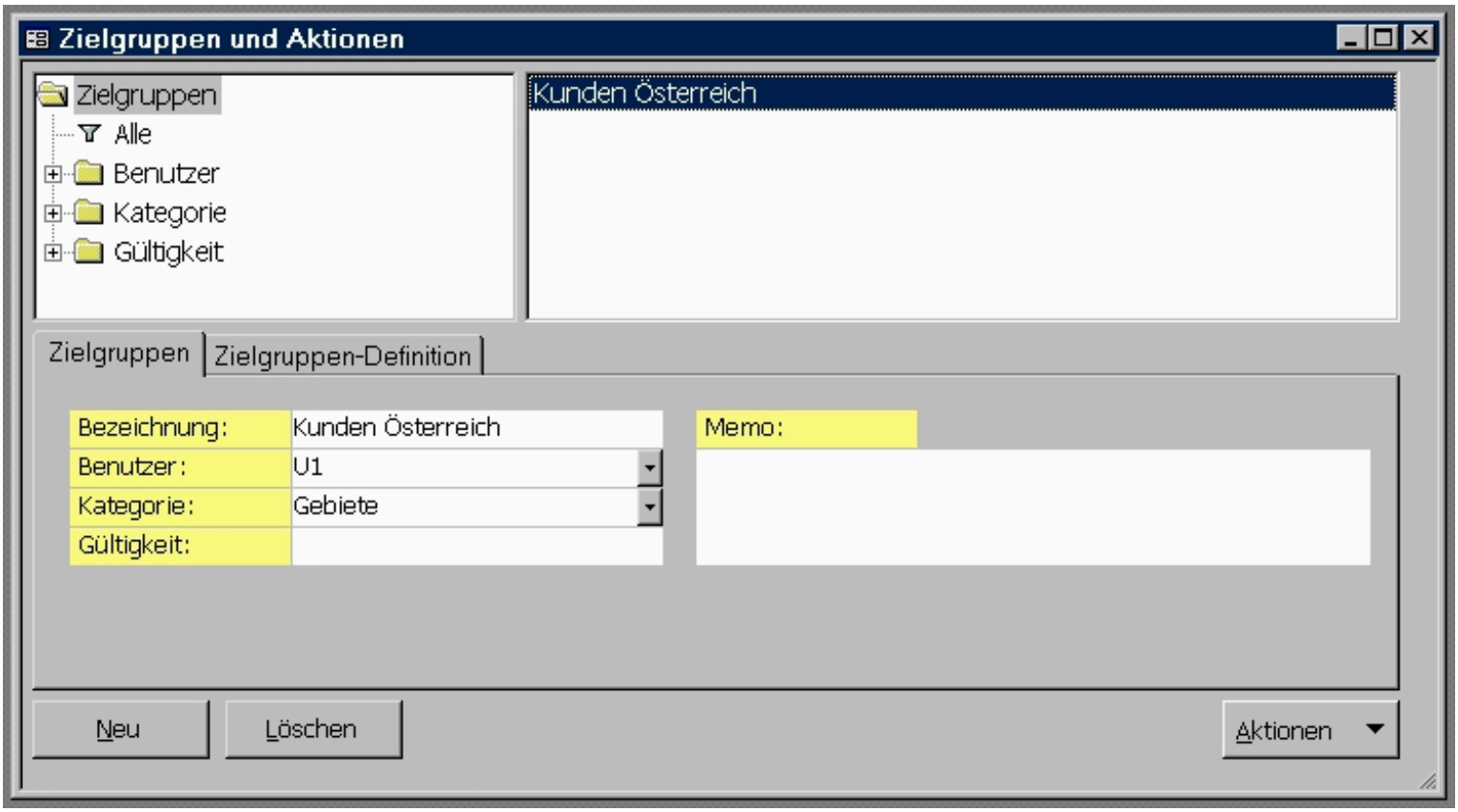


Abbildung 7-48

Im Register "Zielgruppen-Definition" können SQL-Bedingungen erzeugt werden, die zu einer entsprechenden Auswahl von Adressen führen (Abbildung 7-49).

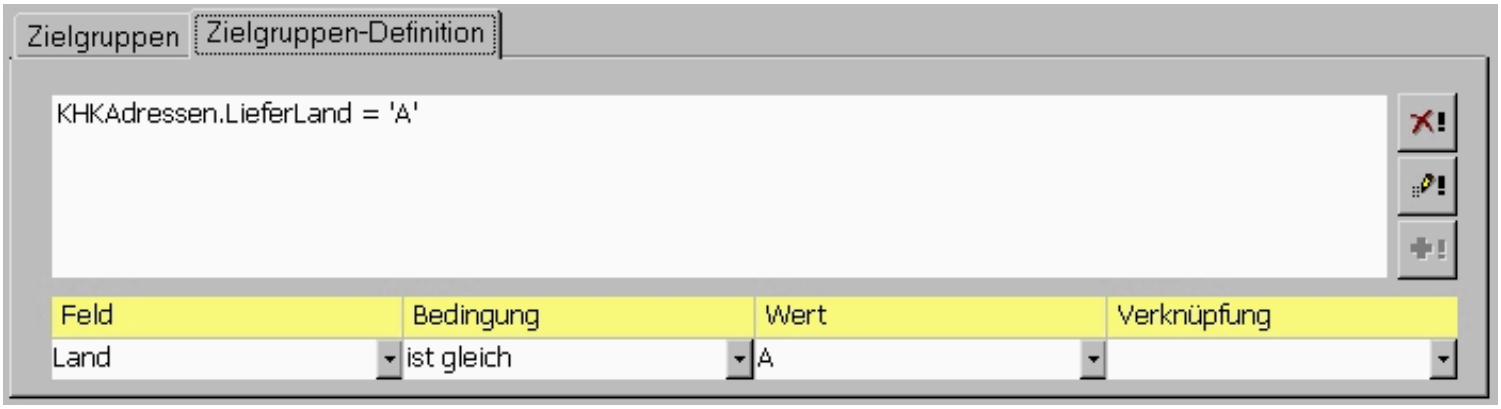


Abbildung 7-49

Mit der Auswahl, die sich aus einer Zielgruppendefinition ergibt, kann man verschiedene Aktionen durchführen (Abbildung 7-50).

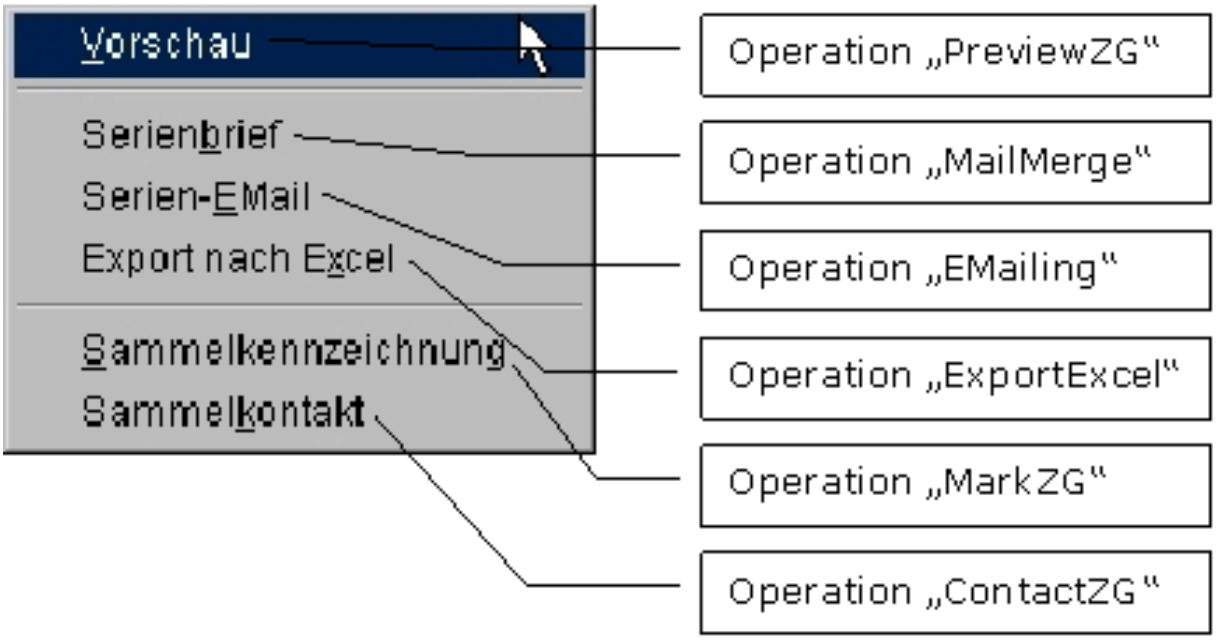


Abbildung 7-50

F380: Die Adressen einer Zielgruppe können über einen Dialog angezeigt werden.

Klasse Zielgruppen

Operation PreviewZG

```
//Vorschau der Adressen einer Zielgruppe
```

```
PreviewZG(in Parameter1 = Zielgruppe.Index)
{
            //Zugehörige Bedingungen auswerten
            GetAddresses(in Bedingungen[1-n], out Parameter2 =
            View(KHKAdressen));
            //Das Ergebnis wird an den passenden Dialog übergeben
                        SHOW Dialog("Vorschau").Data(in Parameter2);
}
```

F390: Mit den Adressen einer Zielgruppe kann in "Microsoft Word" ein Serienbrief erzeugt werden.

Klasse Zielgruppen

Operation MailMerge

```
//Erzeugung eines Serienbriefs
MailMerge(in Parameter1 = Zielgruppe.Index)
{
            //Zugehörige Bedingungen auswerten
            GetAddresses(in Bedingungen[1-n], out Parameter2 =
            View(KHKAdressen));
            //Prüfen ob Word geöffnet ist
            IF NOT OPEN("Microsoft Word")
            THEN
            OPEN("Microsoft Word");
            END IF;
```

```
                    //Über den Datei-Dialog wird ein Word-Dokument ausgewählt
                    //und bekommt die entsprechenden Daten übergeben

                    Word.ActualDocument = Windows.FileDialog.File;

          Word.ActualDocument.MailMergeData = Parameter2;

          //Aktion erzeugen

          Aktionen.Index = NEW;

}
```

F400: Mit den ausgewählten Adressen kann ein Sammel-Emailing durchgeführt werden.

Klasse Zielgruppen

Operation EMailing

```
//Erzeugung eines Serienbriefs

EMailing(in Parameter1 = Zielgruppe.Index)

{

          //Zugehörige Bedingungen auswerten

          GetAddresses(in Bedingungen[1-n], out Parameter2 =

          View(KHKAdressen));

          //Betreffzeile und Text ermitteln

          Parameter3 = INPUT("Betreff?");

          Parameter4 = INPUT("Text?");

          //Die einzelnen Adressen abarbeiten

FOR EACH Record IN Parameter2

//Erzeugung der Microsoft Outlook Objekte

                    Create(out Outlook.EMail);
```

```
Outlook.EMail.To = KHKAdressen.EMail;
Outlook.EMail.Subject = Parameter3;
Outlook.EMail.Body = Parameter4;
//EMail versenden
            Outlook.Email.Send;
            END EACH;
            //Aktion erzeugen
            Aktionen.Index = NEW;
}
```

F410: Die ausgewählten Adressen können in eine Excel-Datei exportiert werden.

Klasse Zielgruppen

Operation ExportExcel

```
//Adressen nach Excel exportieren
ExportExcel(in Parameter1 = Zielgruppe.Index)
{
            //Zugehörige Bedingungen auswerten
            GetAddresses(in Bedingungen[1-n], out Parameter2 =
            View(KHKAdressen));
            //Adressen in Excel-Datei exportieren
            Create Excel.File(Windows.FileDialog.File, Data =
            Parameter2);
            //Aktion erzeugen
```

```
1
            Aktionen.Index = NEW;
}
```

F420: Die benutzerdefinierten Felder der ausgewählten Adressen können auf einen Wert gesetzt werden.

Klasse Zielgruppen

Operation MarkZG

```
//Kennzeichnung der Adressen
MarkZG(in Parameter1 = Zielgruppe.Index)
{
            //Zugehörige Bedingungen auswerten
            GetAddresses(in Bedingungen[1-n], out Parameter2 =
            View(KHKAdressen));
            //Feld auswählen
            Parameter3 = INPUT(DropDown(KHKAdressen.UserFields);
            //Wert ermitteln
            Parameter4 = INPUT("Wert?");
            //Die einzelnen Adressen abarbeiten
FOR EACH Record IN Parameter2
//Wert setzen
            KHKAdressen.Fields(Parameter3) = Parameter4;
            END EACH;
            //Aktion erzeugen
            Aktionen.Index = NEW;
```

1

```
}
```

F430: Für die ausgewählten Adressen kann ein Sammelkontakt erzeugt werden.

Klasse Zielgruppen

Operation ContactZG

```
//Adressen nach Excel exportieren
ContactZG(in Parameter1 = Zielgruppe.Index)
{
            //Zugehörige Bedingungen auswerten
            GetAddresses(in Bedingungen[1-n], out Parameter2 =
            View(KHKAdressen));
            //Erfassungsdialog für Kontakte anzeigen
            SHOW Dialog("Kontaktdetails");
            MESSAGE ("Sammelkontakt speichern?");
IF MESSAGE.Respond = YES
            THEN
                        FOR EACH Record IN Parameter2
//Für jede Adresse einen Kontakt erzeugen
            Kontakte.Index = NEW;
            Kontakte.Adresse = Parameter2.Fields(Adresse);
            Kontakte.Data = Dialog("Kontaktdetails").Data;
                        END EACH;
END IF;
```

```
1
            //Aktion erzeugen

            Aktionen.Index = NEW;

}
```

Im Dialog "Aktionen" (Abbildung 7-51) werden alle Aktionen angezeigt, die mit einer Zielgruppe erzeugt worden sind. Man kann über den Browser nach verschiedenen Kriterien auswählen. Wählt man in der rechten Spalte des Browsers eine Aktion aus, werden in der Liste die zugehörigen Kontakte angezeigt.

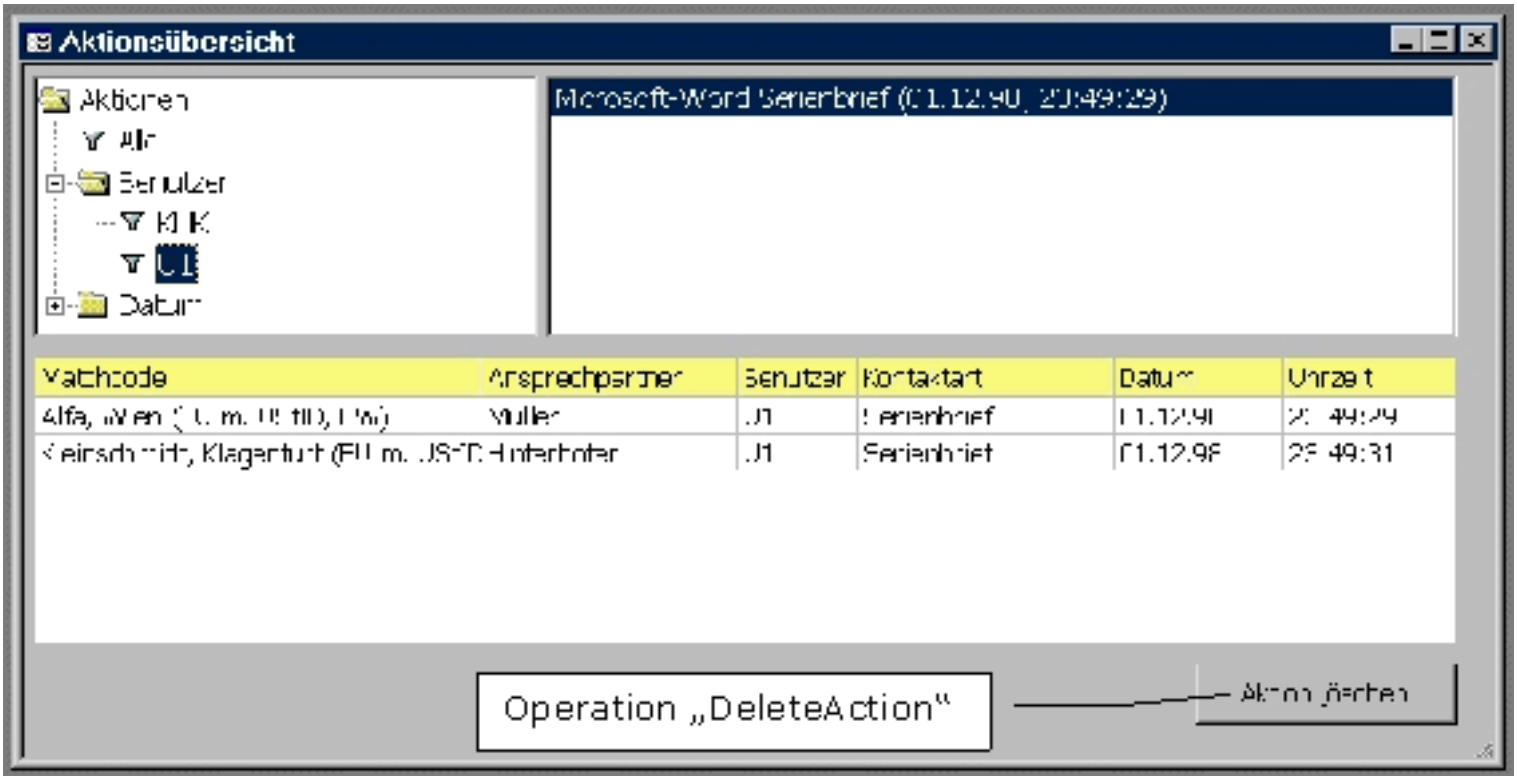


Abbildung (7-51)

F270: Die Aktionen und die zugehörigen Kontakte können nach einer Sicherheitsabfrage gelöscht werden.

Klasse Aktionen

Operation DeleteAction

```
//Aktion löschen

DeleteAction(in Parameter1 = Aktionen.Index)

{
```

```
            MESSAGE ("Aktion löschen?");

IF MESSAGE.Respond = YES

            THEN

                        Aktionen.Delete(in Parameter1);

                        //Zugehörige Kontakte löschen

                        Kontakte.Delete(Condition[Kontakte.Aktion =

                        Parameter1]);

END IF;

}
```

Im Dialog "Projekte" (Abbildung 7-52) können Projekte angelegt, bearbeitet und gelöscht werden. Projekte dienen der Gruppierung von Kontakten. Über den Button "Kontakte" werden alle zugehörigen Kontakte im Dialog "Vorgangsauskunft" angezeigt (Abbildung 7-46).

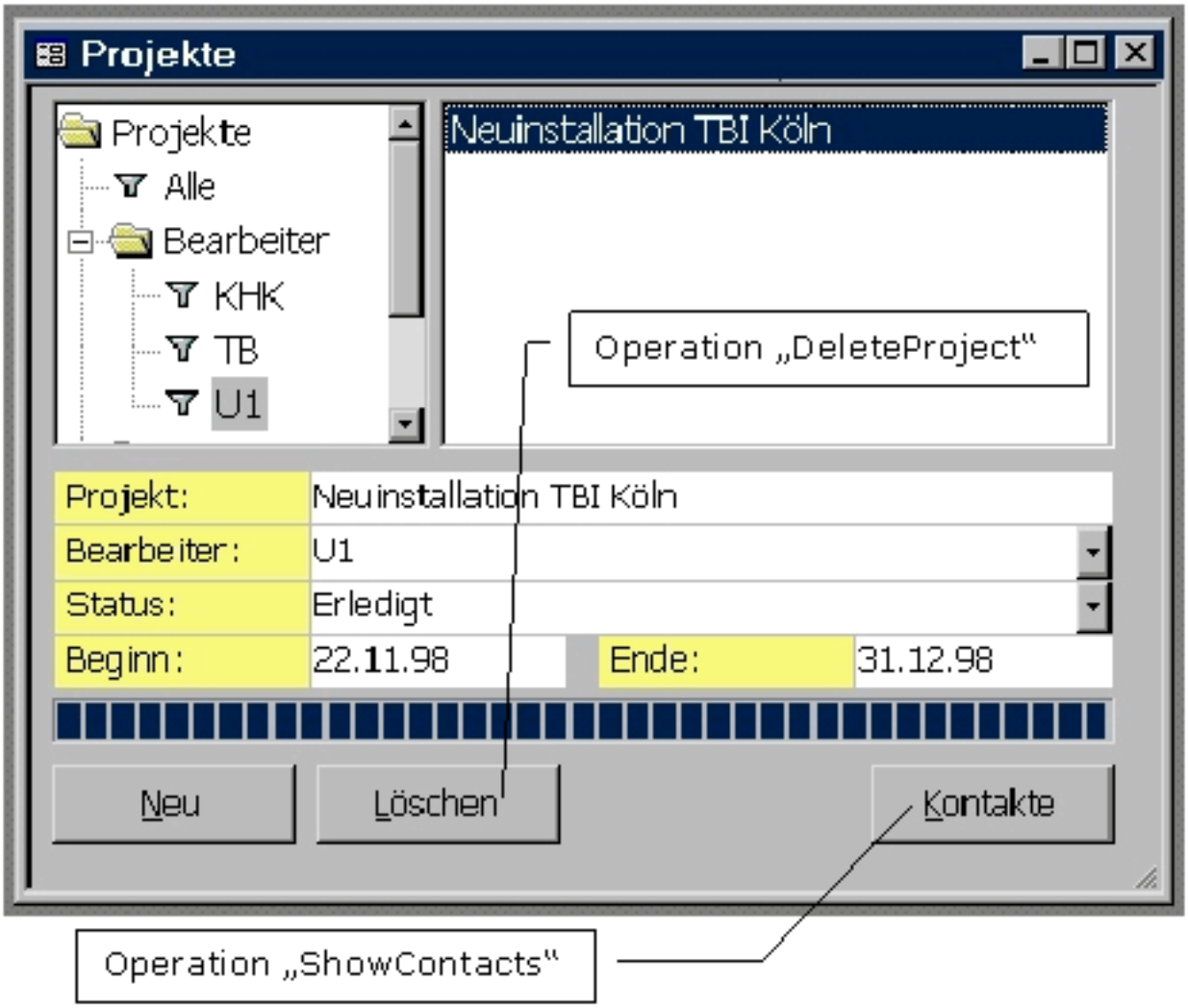


Abbildung 7-52

F310: Soll ein Projekt gelöscht werden, so können nach einer Warnmeldung auch alle zugehörigen Kontakte gelöscht werden.

Klasse Projekte

Operation DeleteProject

```
//Projekt löschen

DeleteProjekt(in Parameter1 = Projekte.Index)

{

            MESSAGE ("Projekt löschen?");

IF MESSAGE.Respond = YES

            THEN

                        Projekte.Delete(in Parameter1);
```

```
                    //Zugehörige Kontakte löschen

                    MESSAGE ("Zugehörige Kontakte löschen?");

IF MESSAGE.Respond = YES

                    THEN

                              Kontakte.Delete(Condition[Kontakte.Projekt =

                              Parameter1]);

END IF;

END IF;

}
```

F290: Alle zugehörigen Kontakte zu einem Projekt können über den Dialog "Vorgangsauskunft" angezeigt werden.

Klasse Projekte

Operation ShowContacts

```
//Kontakte anzeigen

ShowContacts(in Parameter1 = Projekte.Index)

{

//Dialog öffnen und entsprechende Daten auswählen

          SHOW Dialog("Vorgangsauskunft").Data(in Projekte.Index);

}
```

Dieses Kapitel beschreibt Möglichkeiten, ein Klassendiagramm, daß aus einer objektorientierten Analyse hervorgegangen ist, auf eine relationale Datenbank umzusetzen. Es geht dabei besonders um die Behandlung von Klassen, komplexen Typen, Beziehungen und des Entwurfs einer Zugriffsschicht, die zwischen der objektorientierten Anwendung und der relationalen Datenbank implementiert werden muß.

8. Abbildung eines Klassendiagramms auf eine relationale Datenbank

Mit den Ergebnissen der objektorientierten Analyse gilt es einen Weg zur Umsetzung in eine relationale Datenbankumgebung zu finden. Abbildung 8-1 zeigt die notwendigen Transformationen.

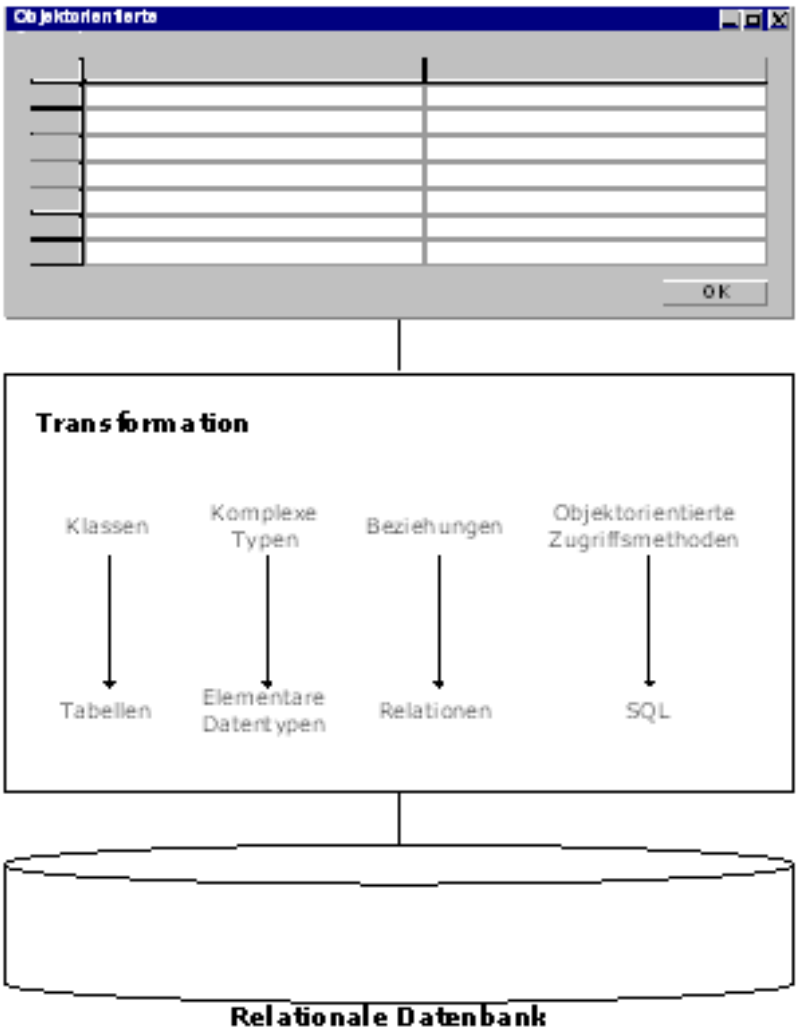


Abbildung 8-1

8.1. Abbildung von Klassen auf Tabellen

Jede Klasse läßt sich auf eine Tabelle der Datenbank abbilden, wobei die Attribute der Klasse den Spalten der Tabelle entsprechen. Um die Vorteile der Objektidentität zu nutzen, erhält jede Tabelle eine Spalte mit einem eindeutigen Schlüssel. Es wird ausdrücklich auf die Möglichkeit eines kombinierten Schlüssels zur Identifikation eines Datensatzes verzichtet. Auf diese Weise kann man jeden Datensatz einer Tabelle als ein Objekt der Tabelle mit eigener Objektidentität verstehen.

Im Beispiel in der Abbildung 8-2 enthält die Spalte Index einen eindeutigen Schlüssel und dient gleichzeitig der Objektidentifizierung.

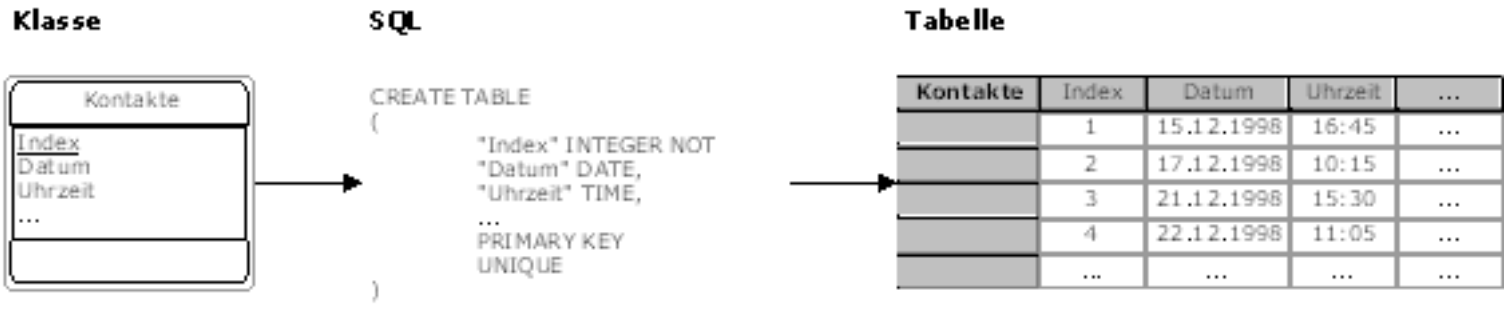


Abbildung 8-2

8.2. Abbildung von komplexen Typen auf elementaren Datentypen

Komplexe Typen in einer objektorientierten Analyse bedeuten für den relationalen Entwurf der Datenbank die Notwendigkeit zur Normalisierung. Die Spaltennamen ergeben sich aus den Komponentennamen des komplexen Typs und können entweder einer Tabelle hinzugefügt werden (Abbildung 8-3[1.]) oder in eine separate Tabelle übertragen werden (Abbildung 8-3[2.]).

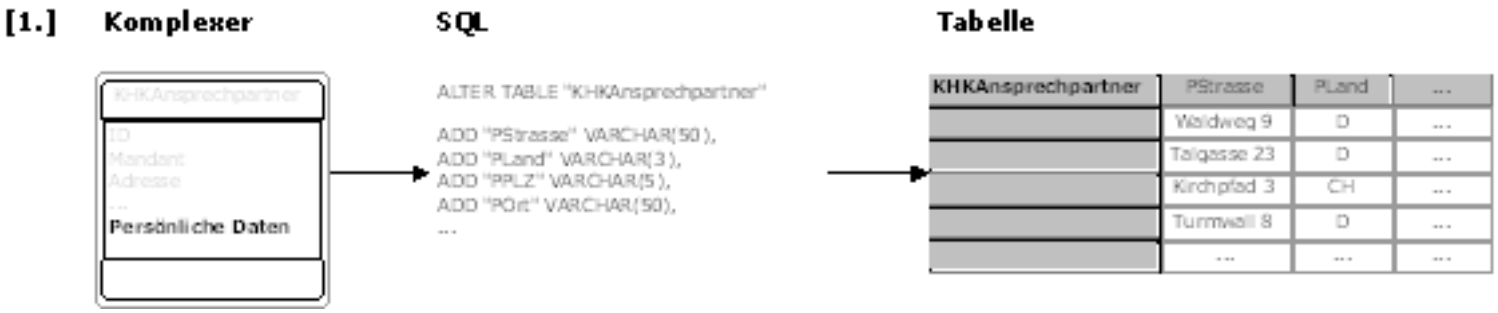


Abbildung 8-3[1.]

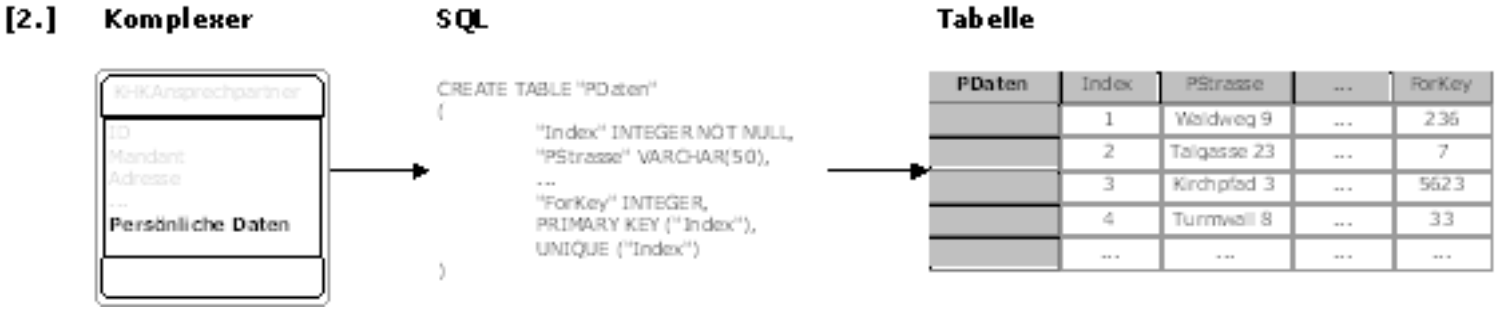


Abbildung 8-3[2.]

Listentypen, deren Elemente keine Beziehungen zu anderen Objekten haben, werden in separaten Tabellen angelegt (Abbildung 8-4).

Abbildung 8-4

8.3. Abbildung von Assoziationen und Aggregationen

Die Abbildung von Assoziationen und Aggregationen auf Relationen richtet sich nach deren Kardinalität.

- 1:1 und 1:m Assoziationen oder Aggregationen werden durch einen Fremdschlüssel in der bezugnehmenden Tabelle definiert (Abbildung 8-5).

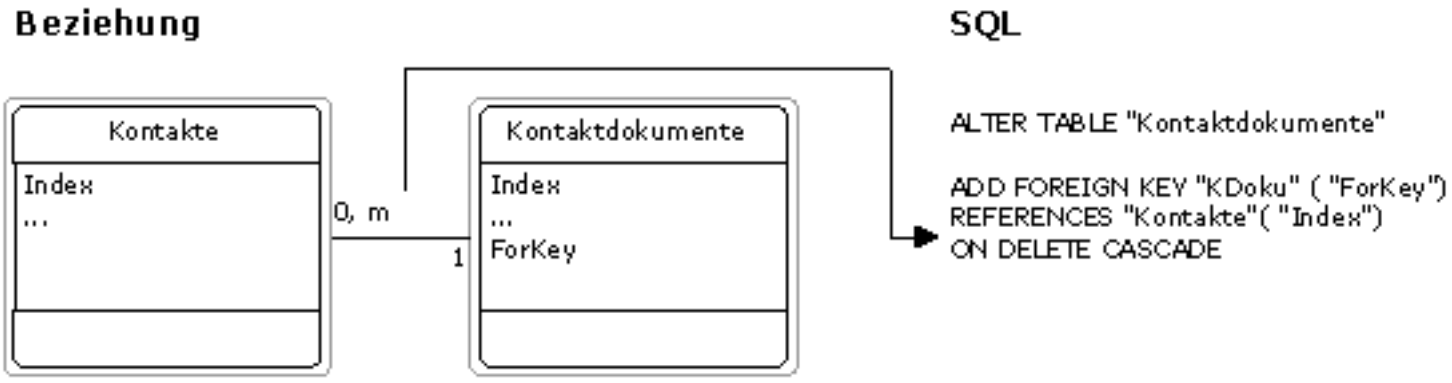


Abbildung 8-5

- Eine m:n Assoziation oder Aggregation wird auf eine eigene Tabelle abgebildet (Abbildung 8-6[1.]). Unter Umständen ist es im Einzelfall jedoch auch sinnvoll, in einer der Tabelle eine Redundanz in Kauf zu nehmen (Abbildung 8-6[2.]).

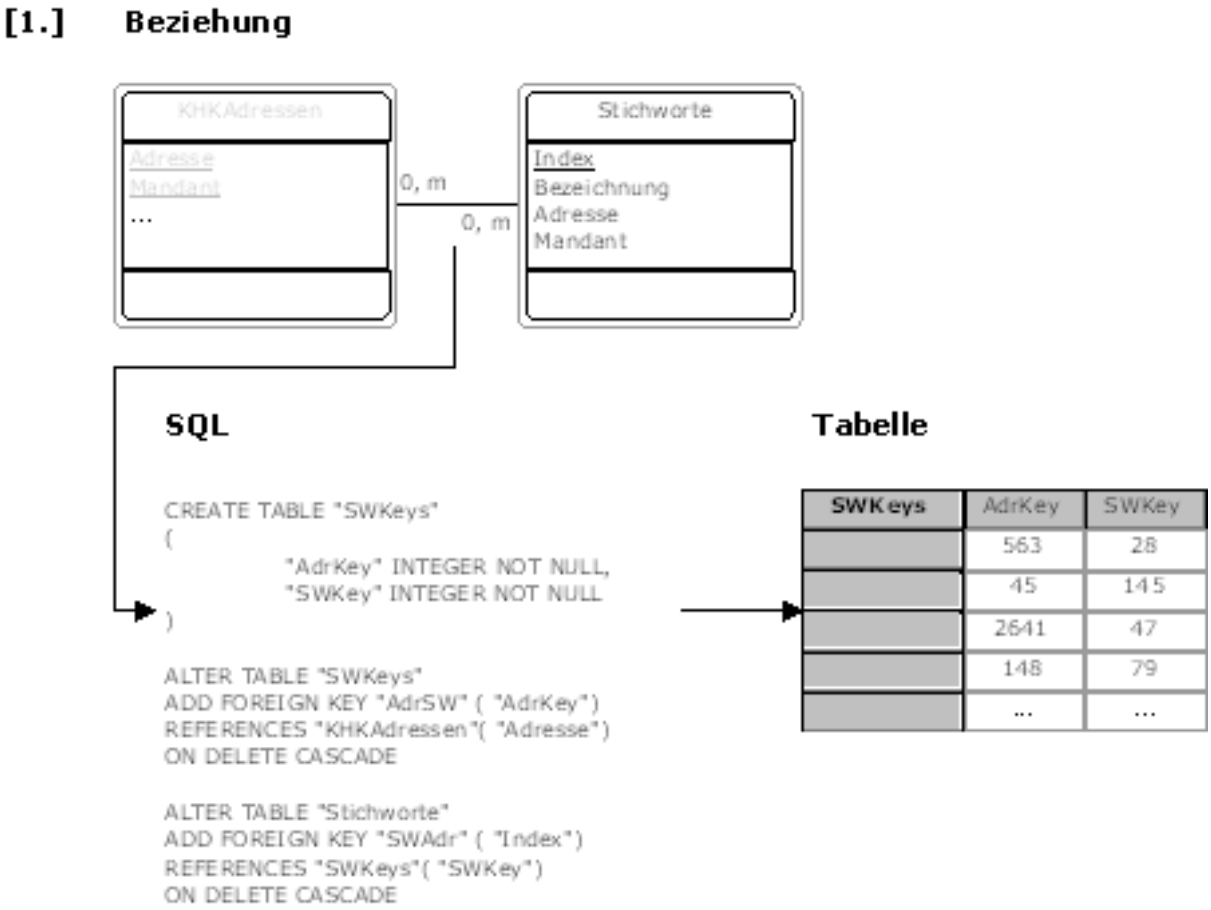


Abbildung 8-6[1.]

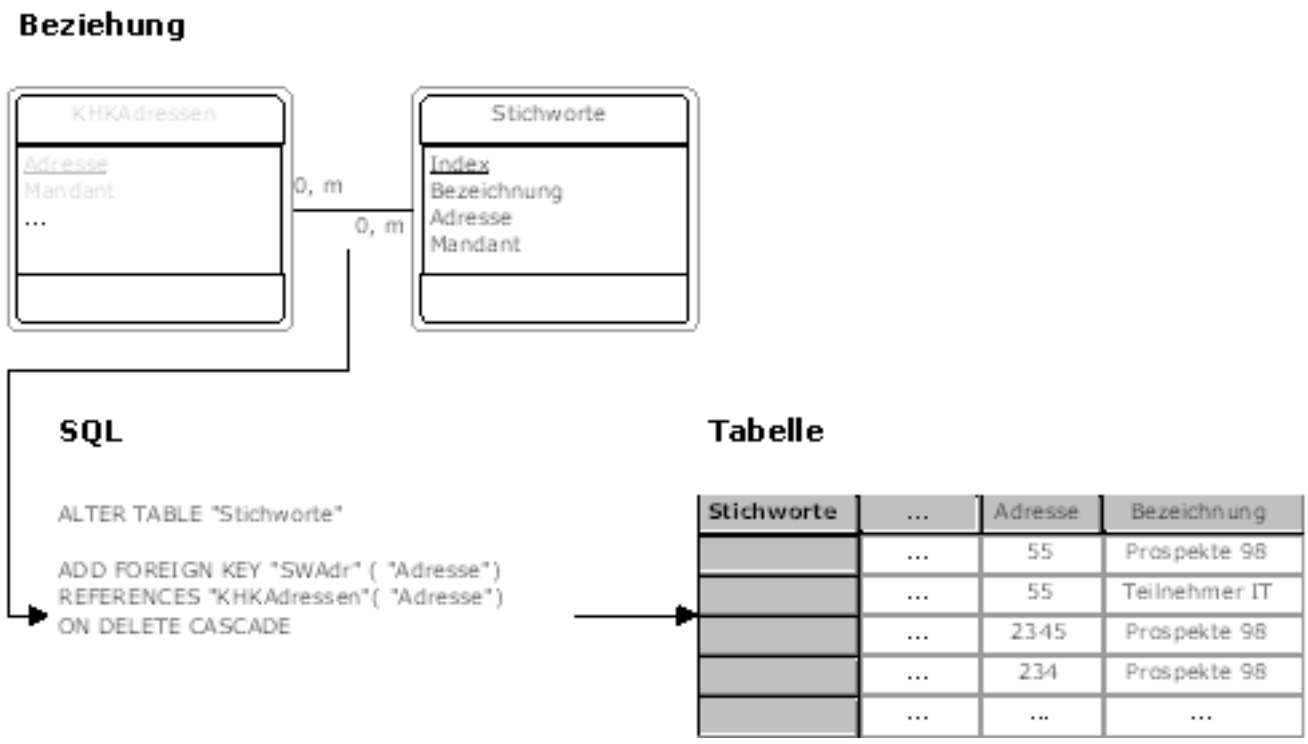


Abbildung 8-6[2.]

8.4. Abbildung von Vererbungen

Bei der Abbildung von Vererbungen ist zunächst zu berücksichtigen, ob es sich um eine Einfachvererbung oder eine Mehrfachvererbung handelt.

Bei einer Einfachvererbung ergeben sich zwei Möglichkeiten der Abbildung auf Tabellen:

- Die Oberklassen und Unterklassen werden auf separate Tabellen abgebildet, die über Relationen miteinander in Verbindung gebracht werden. Das hat den Vorteil, daß gleichzeitig das Datenbankschema minimiert wird, erfordert aber den Zugriff auf mehrere Tabellen, um einen Datensatz auszulesen.

- Alle Oberklassenattribute werden in jede Unterklassen-Tabelle übernommen beziehungsweise alle Unterklassenattribute werden in die Oberklassen-Tabelle übernommen. Dies führt unter Umständen zu Redundanzen, ist im Einzelfall für einen effizienten Umgang mit den Datenobjekten aber durchaus hinzunehmen.

Bei einer Mehrfachvererbung bestehen ebenfalls zwei Möglichkeiten zur Abbildung auf Tabellen:

- Disjunkte Oberklassen und alle Unterklassen werden auf eine Tabelle abgebildet.

- Alle Oberklassen, alle Unterklassen und die Vererbungsstruktur werden in separate Tabellen übertragen und relational in Beziehung gesetzt.

8.5. Behandlung der objektorientierten Zugriffsmethoden

Die Zugriffe einer objektorientierten Anwendung auf eine relationale Datenbank können durch verschiedene Strukturen realisiert werden. Allgemein kann man die notwendigen Schritte mit Hilfe eines Schichtenmodells darstellen (Abbildung 8-7).

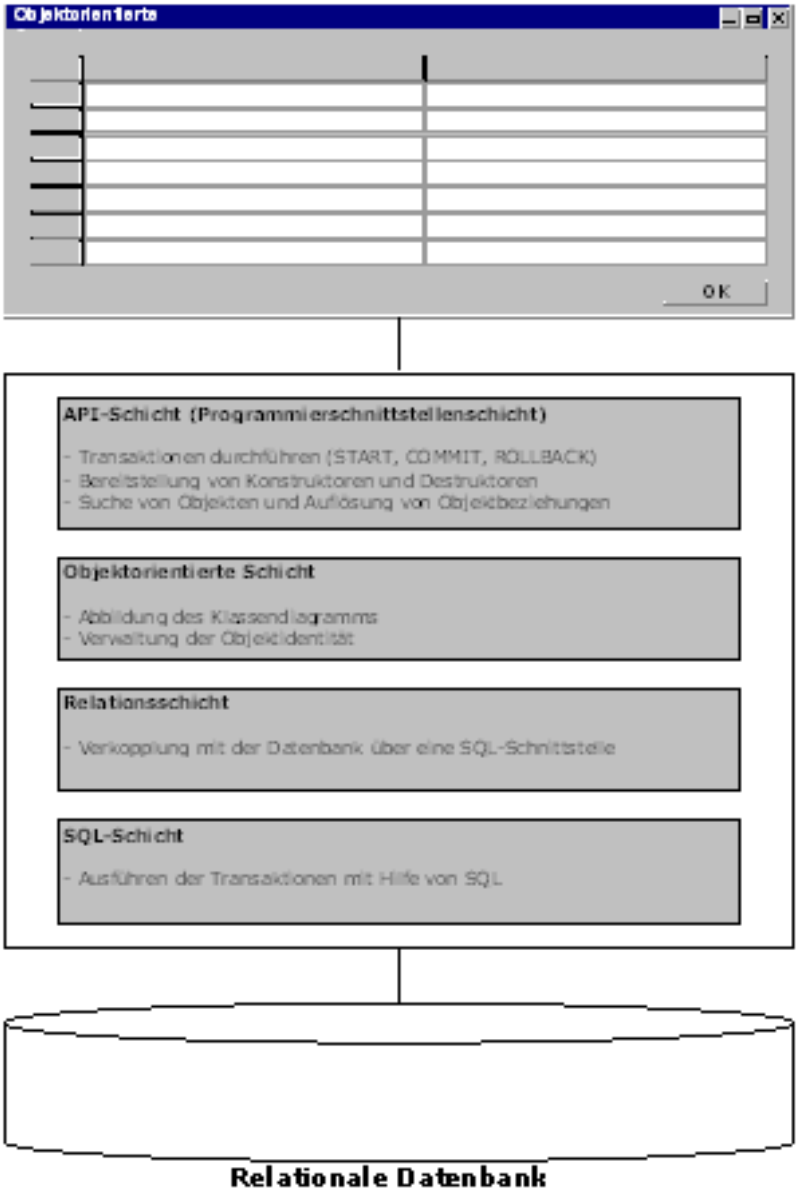


Abbildung 8-7

In der "Windows-Welt" hat sich heute ODBC (Open Data Base Connectivity) als Standardschnittstelle zu relationalen Datenbanken durchgesetzt. Im folgenden Kapitel wird darauf näher eingegangen.

A

Abschließend wird anhand von Beispielen die Implementierung erläutert. Accesss 97 wird als Entwicklungsumgebung vorgestellt, und es wird auf die Bedeutung von ODBC für die Datenbankprogrammierung unter Windows-Sytemen eingegangen.

9. Die Implementierung

Die Ergebnisse der Entwurfsphase hängen in starkem Maße von der verwendeten Arbeits- und Entwicklungsumgebung ab. Die im vorigen Kapitel dargestellten Zugriffsschichten müssen dementsprechend mehr oder weniger in die Implementierung mit einbezogen werden.

Hier wird der Fall beschrieben, daß die Zugriffe auf die relationale Datenbank über die ODBC-Schnittstelle geschehen, und die Anwendung unter "Microsoft Access 97" läuft und mit VBA (Visual Basic for Applications) programmiert wird. Als relationale Datenbank wird "Sybase SQL Anywhere 5.0" verwendet.

9.1. ODBC

ODBC (Open Database Connectivity) ist eine von Microsoft definierte Programmierschnittstelle für den Zugriff von Programmen auf Datenbanken mittels SQL. Für den Programmierer ist ODBC eine Funktionsbibliothek, bei deren Verwendung es nicht notwendig ist, das Programm mit einem Precompiler zu übersetzen oder eine neue Sprache wie Embedded SQL zu lernen. Weitere Vorteile

von ODBC sind die Unterstützung durch mehrere Hersteller auf mehreren Plattformen und die Transparenz bezüglich der Netzverbindung.

Die Architektur von ODBC besteht aus vier Komponenten:

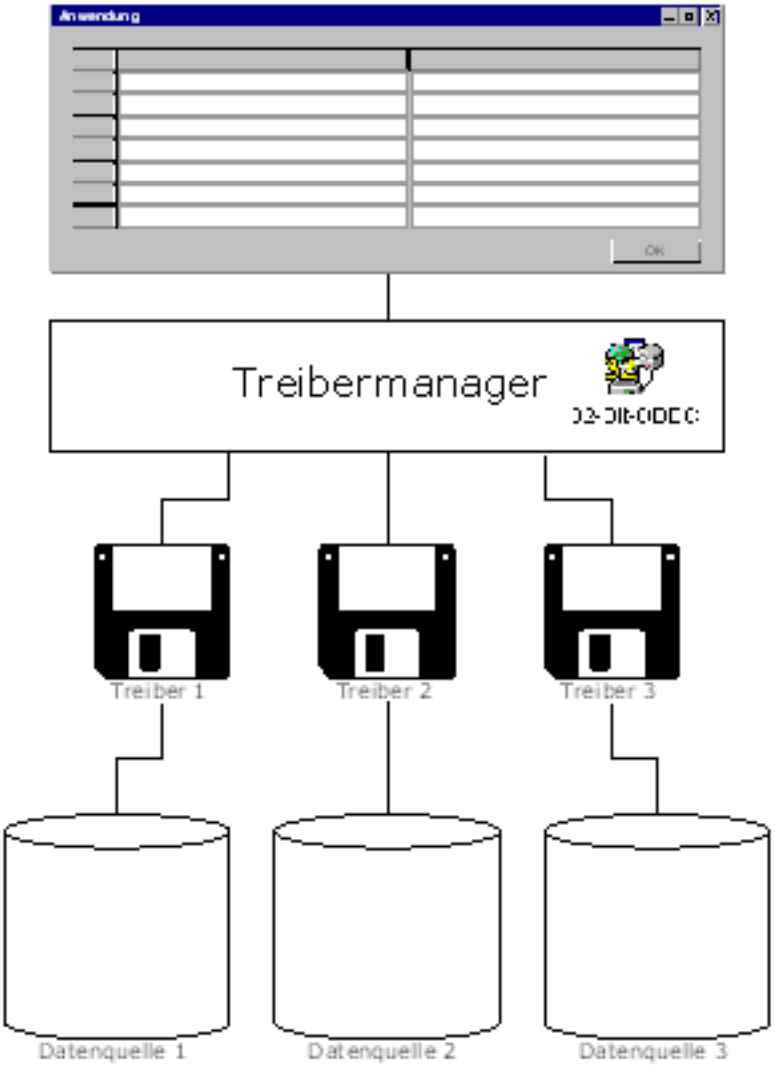


Abbildung 9-1

- Das Programm, das die Daten bearbeitet und über ODBC-Aufrufe SQL-Anweisungen an die Datenbank schickt.
- Den Treiber-Manager, der den entsprechenden Datenbanktreiber bei Bedarf nachlädt.
- Den Datenbanktreiber, der die ODBC-Aufrufe bearbeitet, SQL-Befehle an das Datenbanksystem schickt und die Ergebnisse zurückgibt.
- Die Datenquelle, die in der Regel ein Datenbanksystem ist.

Eine Anwendung kann somit mit denselben Befehlen Daten von beliebigen Datenbanken (für die es einen ODBC-Treiber gibt) auslesen, ändern und löschen. Diese Unabhängigkeit bezeichnet man als *Interoperabilität* (die Anwendung kann mit beliebigen Datenbanken zusammenarbeiten).

A

ODBC definiert bestimmte Konformitätsebenen (Conformance Level) in Bezug auf die unterstützte ODBC API und die unterstützte SQL-Grammatik.

Die Grundmenge der ODBC-Funktionen entspricht den Funktionen in der X/Open und SQL Access Group Call Level Interface-Spezifikation und umfaßt:

- Anlegen und Freigeben von Hilfszeigern (Handles) für das Environment, für Verbindungen und Anweisungen
- Aufbauen der Verbindungen zu Datenquellen
- Vorbereiten und Ausführen von SQL-Anweisungen, direktes Ausführen von SQL-Anweisungen
- Zuweisen von Speicherbereichen für die Datenkommunikation (Parameter und Ergebnismenge)
- Auslesen der Typinformation über die Datensätze der Ergebnismenge einer SQL-Anweisung und Auslesen der Datensätze selber
- Verwalten von Transaktionen (Commit und Rollback)
- Abfragen der Fehlerinformation

Die erste Konformitätsebene (Level 1 API) umfaßt zusätzlich:

- Aufbauen der Verbindung mit treiberspezifischen Dialogboxen
- Abfragen und Setzen von Anweisungs- und Verbindungsoptionen
- Schrittweises Senden von Parameterwerten (für lange Daten)
- Schrittweises Empfangen von Ergebnissen (für lange Daten)
- Erfragen von Kataloginformationen (über Tabellen, Spalten von Tabellen, spezielle Spalten und Statistiken)
- Abfragen von Fähigkeiten des Datentreibers und der Datenquelle wie unterstützte Datentypen, skalare Funktionen und ODBC-Funktionen

Die zweite Konformitätsebene (Level 2 API) umfaßt zusätzlich:

- Auflisten der Informationen über eine Verbindung und über verfügbare Datenquellen

A

- Blockweises Senden von Parameterwerten
- Abfragen der Anzahl der Parameter einer SQL-Anweisung und Erfragen der Typinformation der Parameter
- Cursor zum datensatzweisen Auslesen der Ergebnismenge
- Abfragen der datenbankspezifischen Form einer SQL-Anweisung
- Erfragen erweiterter Kataloginformation (über Privilegien, Schlüssel und Prozeduren)
- Aufrufen einer DLL zur Datenkonversion

Die minimale SQL-Grammatik umfaßt:

- Befehle zur Datendefinition: CREATE TABLE und DROP TABLE
- Befehle zur Datenmanipulation: einfache Form des SELECT, INSERT, UPDATE (mit Suchbedingung) und DELETE (mit Suchbedingung)
- einfache Ausdrücke (wie zum Beispiel A > B + C)
- die Datentypen CHAR und VARCHAR oder LONG VARCHAR

Von den meisten Datenbanktreibern werden zusätzlich angeboten:

- weitere Befehle zur Datendefinition: ALTER TABLE, CREATE INDEX, DROP INDEX, CREATE VIEW, DROP VIEW, GRANT und REVOKE
- erweiterter Befehl zur Datenmanipulation: SELECT mit GROUP und HAVING
- komplexere Ausdrücke wie Unterabfragen und mengenwertige Funktionen wie SUM und MIN
- weitere Datentypen: DECIMAL, NUMERIC, SMALLINT, INTEGER, REAL, FLOAT und DOUBLE PRECISION

Die vollständige SQL-Grammatik umfaßt schließlich noch:

- weitere Befehle zur Datenmanipulation: äußerer Verbund (OUTER JOIN), positioniertes UPDATE, positioniertes DELETE, SELECT FOR UPDATE und Vereinigungen von Ergebnismengen
- skalare Funktionen wie SUBSTRING, ABS und Konstanten für die Datentypen Date, Time

und Timestamp in Ausdrücken

- weitere Datentypen: BIT, TINYINT, BIGINT, BINARY, VARBINARY, LONG VARBINARY, DATE, TIME und TIMESTAMP

- Prozeduraufrufe

Die meisten Funktionen der ODBC-Schnittstelle lassen sich in einfacher Weise auf die Konstrukte von Embedded SQL abbilden.

9.2. Microsoft Access 97

Die Datenbankanwendung Access 97 von Microsoft ist Datenbank und Entwicklungsumgebung in Einem. In den einzelnen Datenbanken, die als MDB (Microsoft Database) Dateien abgespeichert werden, können sich Tabellen, Abfragen, Formulare, Berichte, Makros und Module befinden (Abbildung 9-2). Eine MDB-Datei ist auch mit einer Laufzeitversion von Access 97 lauffähig und kann so unabhängig von einem installierten Access ausgeliefert werden.

Tabellen können direkt in der Access Datenbank vorhanden sein, oder es handelt sich um verknüpfte Tabellen, die über spezielle Treiber angesprochen werden. Optisch erkennt man das an verschiedenen "Icons" für die Tabellentypen. Unter der Eigenschaft "Beschreibung" einer solchen verknüpten Tabelle befindet sich der sogenannte "Connection-String". Beispielsweise bedeutet "ODBC;DSN=Vis1;TABLE=DBA.KHKAdressen", daß es sich um eine ODBC-Verknüpfung handelt mit der DSN (Datasource Name) "Vis1" bezogen auf die Tabelle "DBA.KHKAdressen". Innerhalb der Access Datenbank werden die Tabellen völlig gleichberechtigt angesprochen, soweit der zugehörige Treiber dies erlaubt.

A

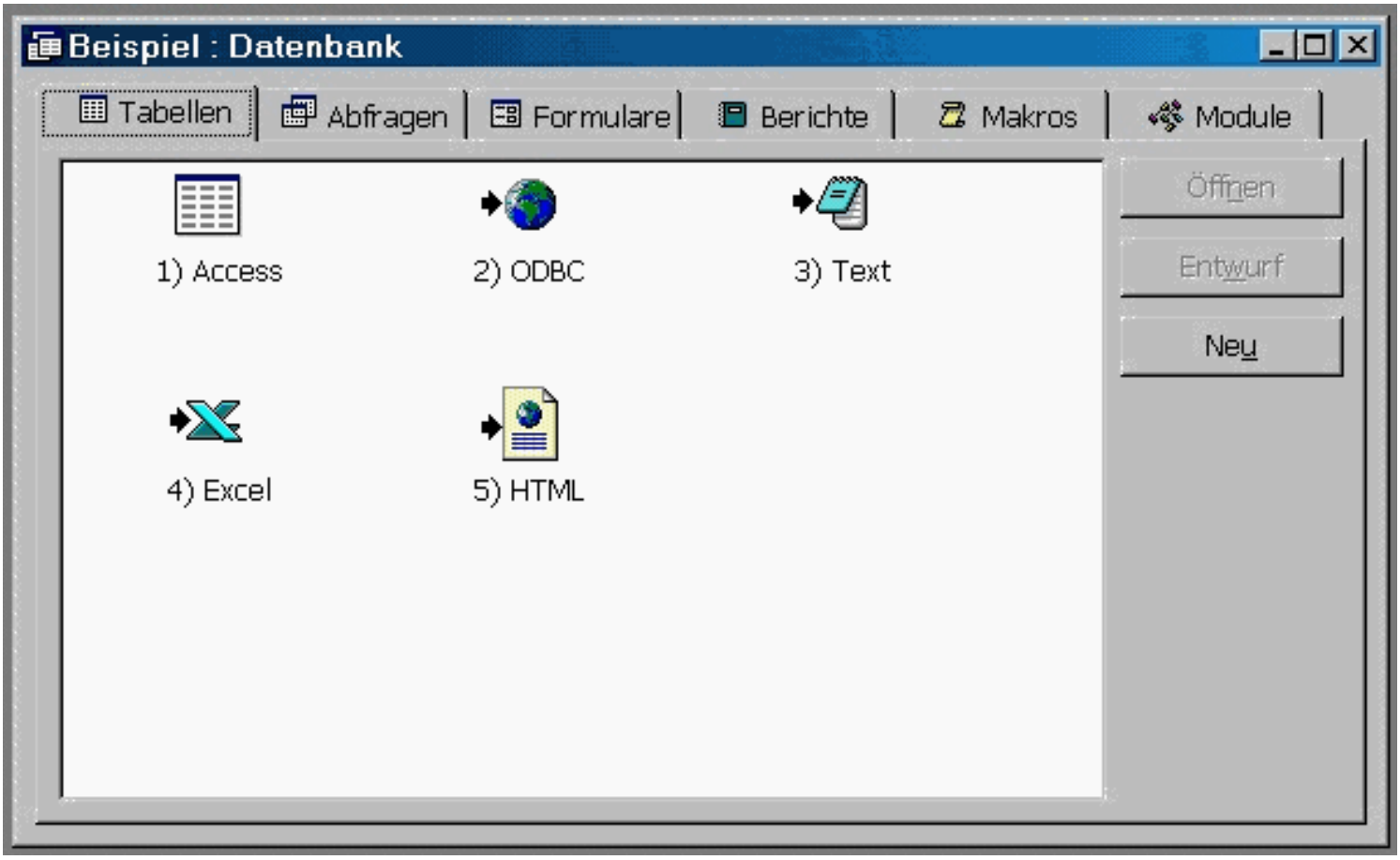


Abbildung 9-2

Abfragen können verschiedene SQL-Abfragen auf beliebige Tabellen oder andere Abfragen enthalten. Sie können dauerhaft in der Datenbank gespeichert werden und stehen dann für Auswertungen oder Aktualisierungen zur Verfügung.

Formulare bilden das Bindeglied zum Programmbenutzer. Mit Hilfe einer grafischen Entwicklungsumgebung können sie vergleichbar zu Visual Basic, Visual C++ oder Delphi entworfen werden. Jedes Formular kann eigenen Programmcode enthalten, der nur während seiner "Existenz" relevant ist. Formulare sind hauptsächlich für den Umgang mit Daten vorgesehen und bieten in der Hinsicht komfortable Funktionen. Ein Formular existiert immer nur als Client-Fenster innerhalb einer Access-Datenbank.

Berichte dienen der grafischen Auswertung von Daten. Sie können ähnlich wie die Formulare innerhalb einer grafischen Entwicklungsumgebung bearbeitet werden und können ebenfalls Programmcode enthalten. Anders als Formulare enthalten Berichte normalerweise ausschließlich Felder, Beschriftungen und Grafiken und bieten sich hauptsächlich für eine Auswertung von Daten am Bildschirm oder über einen Drucker an.

Makros beinhalten von Access angebotene Funktionen. Sie dienen zum Beispiel dazu Tastenkombinationen zur Verfügung zu stellen, oder können als Grundlage für definierte Menüs und Symbolleisten benutzt werden.

Module benötigt man, um Funktionen global in der Datenbank verwenden zu können. Sie bestehen

A

lediglich aus Programmcode.

9.3. Erzeugung der Tabellen und Relationen

Mit den Ergebnissen aus Kapitel 8 kann man die einzelnen Klassen und Assoziationen auf verschiedene Arten auf eine relationale Datenbank umsetzen. Hier wird ein Beispiel gezeigt, wie mit Hilfe des Datenbank-Tools "Interactive SQL" von Sybase und einer SQL-Skript-Datei eine solche Umsetzung aussehen kann.

Mit Hilfe der Benutzer-ID und des Paßwortes wird zunächst eine Verbindung zu einer Datenbank hergestellt. Danach wird die SQL-Skript-Datei geöffnet und ausgeführt. Anhand eines Statistik-Fensters kann der Erfolg der SQL-Anweisungen überprüft werden.

Beispiel:

```
CREATE TABLE "Kontakte"

(

"Index" INTEGER NOT NULL DEFAULT AUTOINCREMENT,

"Datum" DATE,

"Uhrzeit" TIME,

"Adresse" INTEGER NOT NULL,

"Mandant" SMALLINT NOT NULL,

"Benutzer" VARCHAR(4),

"Bezeichnung" VARCHAR(50),

"Memo" LONG VARCHAR,

"Text" LONG VARCHAR,

"Ansprechpartner" VARCHAR(50),

"Kontaktart" VARCHAR(50),

"Projekt" VARCHAR(50),

"Aktion" Integer,
```

```
"KommentarBearb" LONG VARCHAR,

"KommentarKunde" LONG VARCHAR,

"Ergebnis" VARCHAR(50),

PRIMARY KEY ("Index"),

UNIQUE ("Index")

)

GO

ALTER TABLE "Kontakte"

ADD FOREIGN KEY "KeyAdresse" ( "Mandant", "Adresse")

REFERENCES "KHKAdressen"( "Mandant", "Adresse")

ON DELETE CASCADE

GO
```

Anhand des Beispiels kann man erkennen, daß schon durch die Erzeugung der Tabellen und Schlüssel wichtige Grundlagen für die Programmierung gelegt werden können.

Aus objektorientierter Sicht ist es notwendig, daß jedes Objekt eine eindeutige Identität besitzt. Für einen Datensatz, den man als Objekt einer Tabelle betrachten kann, heißt das, daß er nicht nur durch einen Primärschlüssel gekennzeichnet wird, sondern zusätzlich eine Identität erhalten muß. Im Beispiel der Tabelle "Kontakte" übernimmt das Feld "Index" beide Funktionen. Der Zusatz "DEFAULT AUTOINCREMENT" sorgt dafür, daß der Programmierer sich nicht um die Verwaltung des Feld "Index" kümmern muß, und die Integer-Werte automatisch hochgezählt werden.

Mit der Einstellung "NOT NULL" für die Felder "Adresse" und "Mandant" sorgt man dafür, daß bei der Erzeugung eines Kontaktes auch ein Wert für die zugehörige Adresse und den Mandanten angegeben werden muß. Auf diese Weise wird die Muß-Beziehung zwischen der Klasse "KHKAdressen" und der Klasse "Kontakte" ausgedrückt und überwacht. Von der Seite des Programmierers ist lediglich darauf zu achten, daß die Felder mit den richtigen Werten belegt werden.

Der mit "ALTER TABLE" erzeugte Fremdschlüssel "KeyAdresse" stellt eine Beziehung zwischen den Tabellen "KHKAdressen" und "Kontakte" her. Durch die Anweisung "ON DELETE CASCADE" werden beim löschen einer Adresse ebenfalls alle zugehörigen Kontakte gelöscht. Würde man das verhindern wollen, kann man die Anweisung "ON DELETE RESTRICT" verwenden. Existieren in diesem Fall zu einer Adresse noch Kontakte, wenn man diese löschen will, reagiert die Datenbank mit

einer Fehlermeldung. Der Programmierer kann solche Fehler auffangen und dem Benutzer geeignete Optionen zur Auswahl anbieten.

9.4. Globale Funktionen als Container für Objektoperationen

Eine Möglichkeit die Operationen einer Klasse zur Verfügung zu stellen, ist mit globalen Funktionen zu arbeiten. Innerhalb von "Microsoft Access" bieten sich hierfür Module an, deren Funktionen innerhalb der Datenbank genutzt werden können, wenn sie als "public" deklariert sind. Entsprechend dem objektorientierten Gedanken kann man beispielsweise für einzelne Tabellen Module anlegen, welche die Dienstleistungen dieser "Klasse" enthalten. Nachfolgend wird ein Beispiel in "Basic" für die Klasse "KHKAdressen" angeführt.

Beispiel:

Die Dienstleistungen der Klasse "KHKAdressen" werden im Modul "KHKAdressenServices" gespeichert.

KHKAdressenServices

```
Public Function GetAdresse(RelatedTable As String, _

ForeignKey As String, RTIndex As String) As Recordset

Dim SQLString As String

SQLString = "SELECT KHKAdressen.* FROM " & RelatedTable & _

            " INNER JOIN KHKAdressen ON " & RelatedTable & "." & _

            ForeignKey & " = KHKAdressen.Adresse WHERE & _

            KHKAdressen.Adresse = " & RTIndex & ";"

Set GetAdresse = CurrentDb.OpenRecordset(SQLString)

End Function

...
```

Die Funktion "GetAdresse" bekommt als Argumente den Namen der bezogenen Tabelle, den Namen des Fremdschlüssels und den Wert des Fremdschlüssels übergeben. Das Ergebnis ist der Datensatz, auf den sich der Fremdschlüssel bezieht, und somit das assoziierte Objekt.

A

9.4. OCX-Elemente als Container für Objektoperationen

In einer Entwicklungsumgebung stellen OCX-Elemente graphische Funktionsbibliotheken dar, die bei der Programmierung in der Fensterfläche plaziert und bearbeitet werden können. Die meisten Eigenschaften und Funktionen eines OCX-Elementes können zur Entwurfszeit manipuliert werden.

Für die Datenbearbeitung und –navigation kann man diese Elemente im Sinne des objektorientierten Ansatzes auf vielfältige Weise nutzen. An dieser Stelle wird ein OCX-Element beschrieben, daß der Navigation in Datensätzen dient.

Zur Entwurfszeit wird das Element auf einer Fensterfläche positioniert (Abbildung 9-3). Seine Eigenschaften (z.B. Höhe, Breite etc.) können über einen Dialog verändert werden. Zusätzlich kann über Quellcode festgelegt werden, was bei einer Benutzeraktion (z.B. Doppelklick) passieren soll.

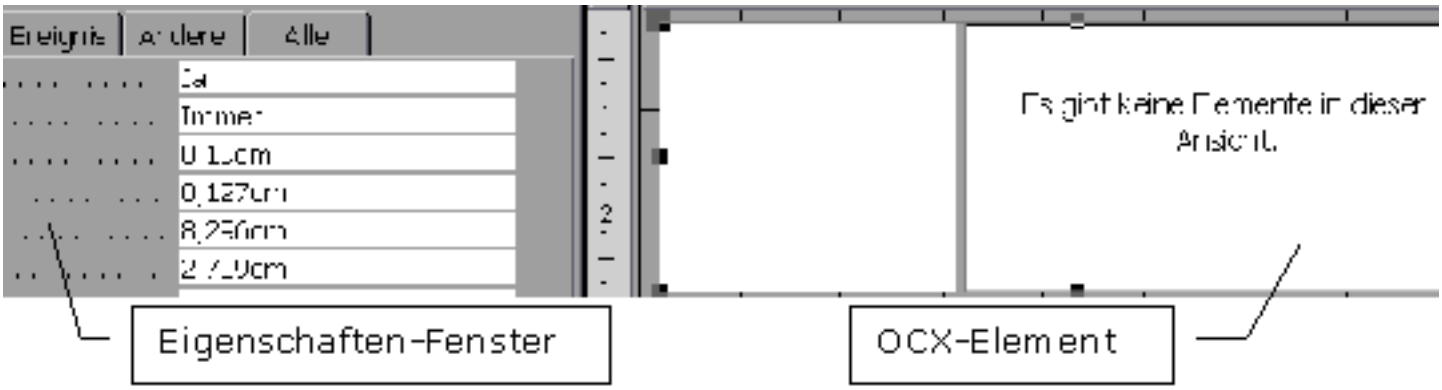


Abbildung 9-3

Der dargestellte Datenbrowser arbeitet hauptsächlich mit Hilfe von Systemtabellen. In diesen Tabellen ist die Struktur der betreffenden Tabellen der relationalen Datenbank mit objektorientierten Begriffen beschrieben. Es handelt sich dabei um eine Tabelle, die die Klasse beschreibt (Abbildung 9-4), eine Tabelle, die gegebenenfalls verbundenen Klassen und Identitäten beschreibt (Abbildung 9-5) und eine Tabelle, die die Auswahlkriterien enthält (Abbildung 9-6).

	Class	Caption	Table	Clause	KeyName	KeyType	Lookup	Autoindex
	Projekte	Projekte	IPVisProjekte	Mandant=$1	Index	0	Projekt	Nein
▶						0		Nein

Abbildung 9-4

	Class	Index	KeyDescription	KeyType	ForeignTable	ForeignTableJoin
	Projekte	0		0		
▶		0		0		

Abbildung 9-5

A

	Class	Index	Parent	Indent	Caption	WhereClause	ItemSource
	Projekte	2000	0	0	Alle	Projekt LIKE '*'	
	Projekte	2001	0	0	Bearbeiter		SELECT DISTINCT Benutzer, Benu
	Projekte	2002	0	0	Status		SELECT DISTINCT Status, Status A
▶	Projekte	2003	0	0	Beginn		SELECT DISTINCT Beginn, Beginn
	Projekte	2004	0	0	Ende		SELECT DISTINCT Ende, Ende AS
*		0	0	0			

Abbildung 9-6

Initialisiert man das OCX-Element beim Öffnen des betreffenden Dialoges mit der Klasse "Projekte", stellt sich das Element wie in Abbildung 9-7 zu sehen dar.

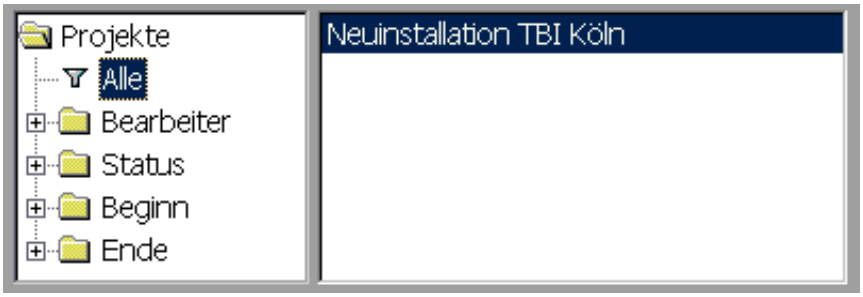


Abbildung 9-7

Die Funktionalität ist vollständig in dem OCX-Element gekapselt. Die Steuerung erfolgt über die Systemtabellen und ist dadurch weitgehend unabhängig von den zugrundeliegenden Daten.

Auf diese Weise lassen sich Klassenstrukturen durch zusätzliche Definition von Regeln sinnvoll in einer Anwendung nutzen, die vom Zugriffsmechanismus ausschließlich mit relationalen Daten arbeitet.

10. Zusammenfassung

Heute findet man im betrieblichen Alltag fast ausschließlich relationale Datenbanken. Auf der anderen Seite hat sich bei der Entwicklung von Software der objektorientierte Gedanke weitgehend durchgesetzt. Ziel der Diplomarbeit war es, eine Methode zu entwickeln, bei der ein Programm vollständig unter objektorientierten Aspekten entwickelt und auf eine relationale Datenbank aufgesetzt wird.

Am Anfang jeder Software steht zunächst eine Idee, oder das Bedürfnis Arbeit einzusparen oder zu erleichtern. Alle nachfolgenden Schritte sollten schon als Prozeß der Softwareentwicklung verstanden werden.

Die Systemanalyse ist der erste und wichtigste Schritt auf dem Weg zum neuen Softwareprodukt. Dabei wird geprüft, welche Anforderungen an das Programm gestellt werden. Die Anforderungen ergeben sich zunächst aus Gesprächen mit den beteiligten Personen, aus ersten Entwurfsdokumenten und gegebenenfalls aus Fachliteratur zu dem betreffenden Thema. Im Verlauf der Analyse wird geklärt, welche Funktionen zu realisieren sind, wie hoch der Aufwand ist und ob ein Zukauf von Komponenten oder Programmen angebracht ist. Meistens ist auch eine Bestimmung von Schnittstellen zu berücksichtigen. Als Ergebnis der Systemanalyse entsteht ein Pflichtenheft, daß für den weiteren Verlauf des Projektes als Grundlage dient, und daß darüber hinaus als Vertragsgrundlage dienen sollte.

Die anschließende objektorientierte Analyse baut auf den Ergebnissen des Pflichtenheftes auf. Sie sollte bemüht sein, sich an den vorgegebenen Begriffen zu orientieren und diese auf ein angemessenes Abstraktionsniveau zu bringen. Die Programmbausteine und -funktionen werden aus dem Pflichtenheft abgelesen und in Diagramme überführt. Die objektorientierte Analyse stellt danach zunächst hauptsächlich die betroffenen Objekte und deren Beziehungen untereinander dar. Um die Funktionen zu verdeutlichen und außenstehende Personen besser in die Entwicklung mit einzubeziehen, kann während der Analyse schon mit ersten Bildschirmmasken gearbeitet werden. Sie sollten nicht unbedingt mit funktionsfähigem Programmcode hinterlegt sein, sondern sollten viel mehr mit Hilfe von Pseudocode die Funktionen verdeutlichen und lesbar machen. Auf diese Weise kann man frühzeitig die Resonanz der Auftraggeber oder Benutzer berücksichtigen und gegebenenfalls auf Änderungswünsche oder Fehlentwicklungen eingehen.

Die Ergebnisse der objektorientierten Analyse sind Ausgangspunkt für die Implementierung des Projektes. Da als Datenbasis aber in den meisten Fällen eine relationale Datenbank verwendet wird, sind vorher noch Maßnahmen zu treffen, um die definierten Klassen, ihre Objekte und die gegenseitigen Beziehungen auf Tabellen abbilden zu können. Ein Objekt, daß in diesem Fall der Datensatz einer Tabelle ist, muß zu jeder Zeit eine eindeutige Identität besitzen. Ein Primärschlüssel wird also in manchen Fällen nicht ausreichen, und die betreffende Tabelle muß um einen eindeutigen Schlüssel, die Objektidentität, ergänzt werden. Die Beziehungen der Analyse werden mit den Mitteln der jeweiligen Datenbank abgebildet (Trigger, Stored Procedures etc.). Dieser Punkt ist entscheidend für die Konsistenz des Datenbestandes und kann den späteren Programmieraufwand wesentlich verringern.

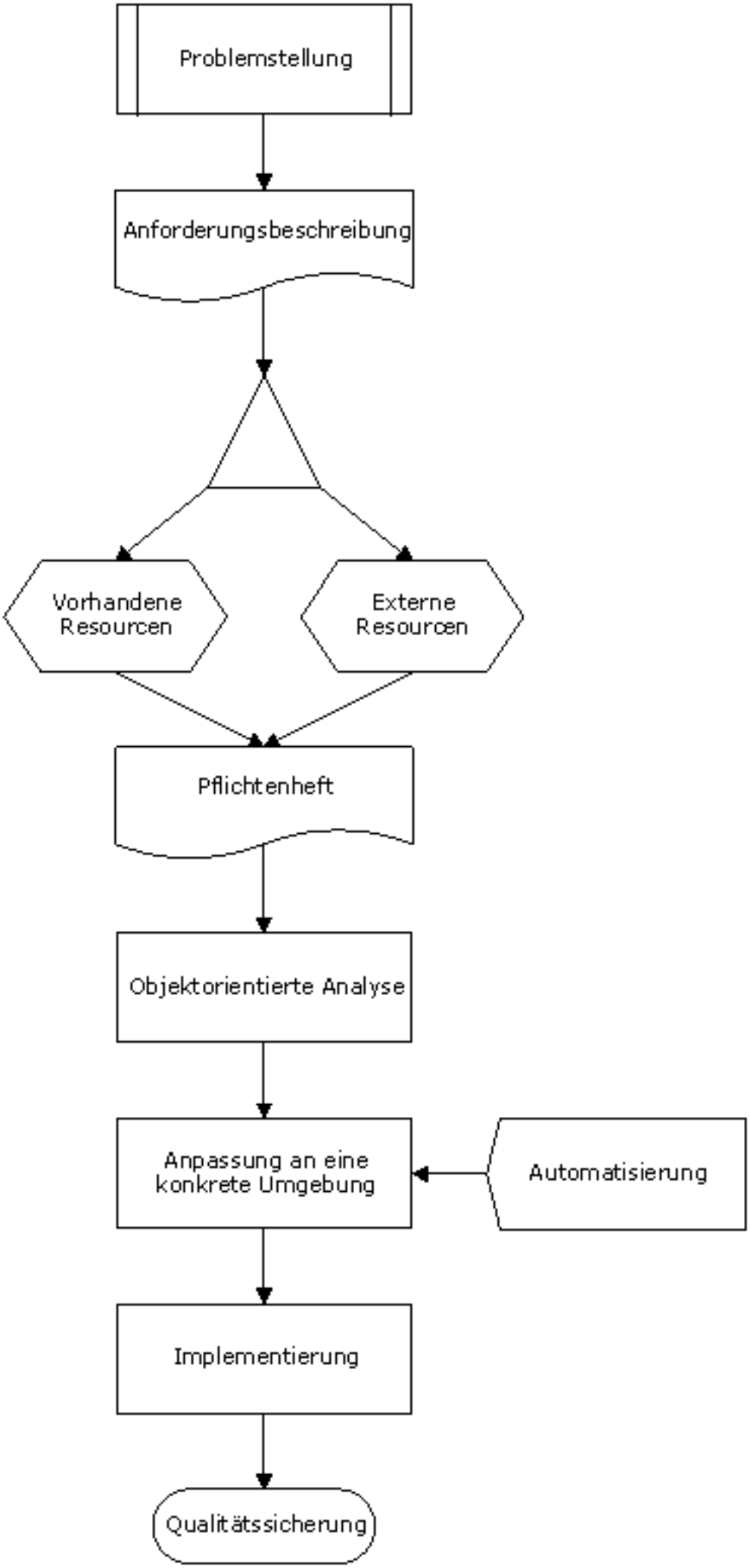

Problemstellung
Anforderungsbeschreibung
Vorhandene Resourcen
Externe Resourcen
Pflichtenheft
Objektorientierte Analyse
Anpassung an eine konkrete Umgebung
Automatisierung
Implementierung
Qualitätssicherung

Abbildung 10-1

Bei der anschließenden Programmierung hängt es stark von der verwendeten Entwicklungsumgebung ab, inwieweit man die Klassenstruktur der relationalen Datenbank ausnutzen kann. Die ODBC-Schnittstelle von Microsoft bietet eine, von vielen Datenbankherstellern unterstützte Schnittstelle, zu relationalen Datenbanken. Wird der Zugriff darauf von der Entwicklungsumgebung unterstützt, ist der Programmierer beim Zugriff auf die Daten lediglich durch den Funktionsumfang des jeweiligen Treibers beschränkt. Er spricht die Datenbankobjekte mit SQL-Befehlen an und kann sie im Programm als lokale Objekte verwenden. Die Objektoperationen kann er in Funktionsbibliotheken kapseln und somit die Dienstleistungen der einzelnen Klassen dem Anwender zur Verfügung stellen.

Abkürzungen

CAS = Computer Aided Selling

DBMS = Datenbankmanagementsystem

DD = Data Dictionary

DL = Definitionssprache

DML = Datenmanipulationssprache

ER-Modell = Entity-Relationship Modell

ML = Manipulationssprache

SQL = Structured Query Language

ODBC = Open Data Base Connectivity

ODL = Object Definition Language

ODMG = Object Database Management Group

OOA = Objektorientierte Analyse

OOD = Objektorientierter Entwurf

OOP = Objektorientierte Entwicklung

OQL = Object Query Language

VIS = Vertriebsinformationssystem

Literaturverzeichnis

Bücher:

- Lehrbuch der Software-Technik, Software-Entwicklung

Helmut Balzert

- Lehrbuch der Software-Technik, Software-Management

Helmut Balzert

- Programmiersprachen für Datenbanken

N. Paton, R. Cooper, H. Williams, P. Trinder

- Datenbank-Management

Ulrich Kracke

- Access 97

Ernst Tiemeyer, Klemens Konopasek

- Marktspiegel: CAS-Standardsoftware zur dezentralen Vertriebs- und Außendienststeuerung mit PC

Dornis, Herzig

Internet:

- http://www.oi.com

Object International

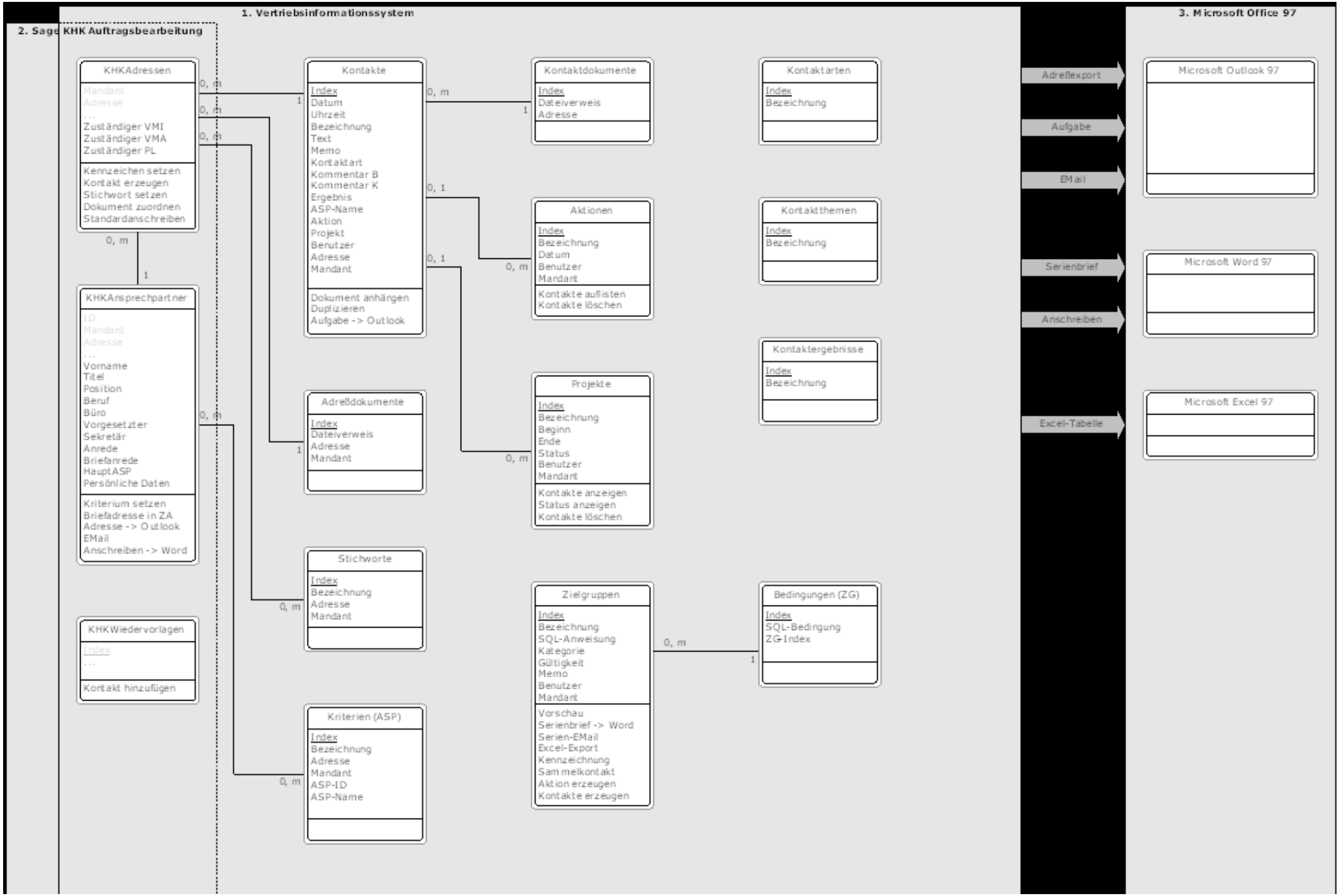
- http://www.khk.de

KHK Software AG

- http://www.microsoft.com

Microsoft Corporation

Anhang


1. Vertriebsinformationssystem
2. Sage KHK Auftragsbearbeitung
3. Microsoft Office 97
KHKAdressen
Mandant
Adresse
...
Zuständiger VMI
Zuständiger VMA
Zuständiger PL
Kennzeichen setzen
Kontakt erzeugen
Stichwort setzen
Dokument zuordnen
Standardanschreiben
Kontakte
Index
Datum
Uhrzeit
Bezeichnung
Text
Memo
Kontaktart
Kommentar B
Kommentar K
Ergebnis
ASP-Name
Aktion
Projekt
Benutzer
Adresse
Mandant
Dokument anhängen
Duplizieren
Aufgabe -> Outlook
Kontaktdokumente
Index
Dateiverweis
Adresse
Kontaktarten
Index
Bezeichnung
Aktionen
Index
Bezeichnung
Datum
Benutzer
Mandant
Kontakte auflisten
Kontakte löschen
Kontaktthemen
Index
Bezeichnung
Kontaktergebnisse
Index
Bezeichnung
KHKAnsprechpartner
ID
Mandant
Adresse
...
Vorname
Titel
Position
Beruf
Büro
Vorgesetzter
Sekretär
Anrede
Briefanrede
HauptASP
Persönliche Daten
Kriterium setzen
Briefadresse in ZA
Adresse -> Outlook
EMail
Anschreiben -> Word
Adreßdokumente
Index
Dateiverweis
Adresse
Mandant
Projekte
Index
Bezeichnung
Beginn
Ende
Status
Benutzer
Mandant
Kontakte anzeigen
Status anzeigen
Kontakte löschen
Stichworte
Index
Bezeichnung
Adresse
Mandant
KHKWiedervorlagen
Index
...
Kontakt hinzufügen
Zielgruppen
Index
Bezeichnung
SQL-Anweisung
Kategorie
Gültigkeit
Memo
Benutzer
Mandant
Vorschau
Serienbrief -> Word
Serien-EMail
Excel-Export
Kennzeichnung
Sammelkontakt
Aktion erzeugen
Kontakte erzeugen
Bedingungen (ZG)
Index
SQL-Bedingung
ZG-Index
Kriterien (ASP)
Index
Bezeichnung
Adresse
Mandant
ASP-ID
ASP-Name
Adreßexport
Aufgabe
EMail
Serienbrief
Anschreiben
Excel-Tabelle
Microsoft Outlook 97
Microsoft Word 97
Microsoft Excel 97
0, m
1
0, 1